The PHP Company

User Guide
Zend Studio 7.x

By Zend Technologies, Inc.

www.zend.com

Disclaimer

The information in this help is subject to change without notice and does not represent a commitment on the part of Zend Technologies Ltd. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of Zend Technologies Ltd.

All trademarks mentioned in this document, belong to their respective owners.
© 1999-2010 Zend Technologies Ltd. All rights reserved.

Zend Studio 7.x User Guide issued April 2010.

Product Version: Zend Studio 7.x
DN: ZS-UG-210410-07-02

Table of Contents

What's New in Zend Studio 7.2 16
Mac OS X 18
Getting Started 19
Quick Start 20
Getting Started 20
Workbench 20
Workspaces 20
Creating a PHP Project 21
Creating a PHP File 21
PHP Debugging 21
Remote Server Explorer (RSE) 22
Source Control 23
PHPUnit 23
Refactoring 24
Perspectives of Interest 24
First Steps for Developing Applications with Zend Studio 25
Install Zend Studio 25
Launch Zend Studio 26
Install Zend Server (optional) 27
Create a PHP project 28
Create PHP Files 28
Create PHP Elements 29
Write Code 30
Rename Elements (Refactoring) 31
Run your application 32
Next Steps 32
Switching from Zend Studio 5.x to Zend Studio 7.x 33
Project/File Creation 33
Debugging 35
Profiling 39
Source Control - Subversion (SVN) 40
FTP Connectivity 43
Database Connection 44
Tunneling 46
Basic Tutorials 47

Creating Projects and Files 48
Creating a PHP Project 48
Creating a PHP File 49
Working with Content Assist 50
Purpose and Usage 50
Activating Content Assist 51
Function Parameter Hints 52
Making Content Assist Elements Available Within the Same Scope 52
Class Type Hints 54
Configuring Content Assist 54
Working with CVS 55
Purpose 55
Adding a CVS Repository 56
Sharing Projects 57
Checking Out Projects from CVS 58
Adding Files to Existing Projects 59
Making Changes, Comparing Changes, and Committing Changes 60
Replacing Files with Older Versions 61
Deleting Files from CVS 62
Working with SVN 63
Purpose 63
Adding an SVN Repository 63
Sharing Projects 64
Checking Out Projects from SVN 65
Adding Files to Existing Projects 65
Making Changes, Comparing Changes, and Committing Changes 66
Replacing Files with Older Versions 68
Deleting Files from SVN 69
Working with the Debugger 70
Purpose and Usage 70
Debugging Local PHP Files (PHP Scripts) 71
Debugging PHP Applications (PHP Web Pages) 78
Working with Refactoring 81
Purpose and Usage 81
Renaming / Moving Files 82
Renaming Elements within a File 83
Working with the Profiler 84

Purpose and Usage 84
Profiling PHP Scripts 84
Profiling PHP Web Pages 89
Working with PHPUnit Testing 91
Purpose and Usage 91
Creating a PHPUnit Test Case 92
Running your PHPUnit Test Case 95
Analyzing Errors 97
Creating and Running a PHPUnit Test Suite 98
Generating PHPUnit Test Reports 99
Working with Zend Server 100
Purpose and Usage 100
Configuring Zend Server Integration 100
Creating a project on a Local Zend Server 102
Diagnosing and Profiling Performance Issues on Zend Server 102
Diagnosing and Debugging Errors On Zend Server 106
Concepts 110
Update Manager 111
PHP Version Support 112
Content Assist 114
Using elements within the same scope 114
Function Parameter Hints 115
Content Assist for Include Statements 115
Class Type Hints 116
Content Assist for Magic Members 117
Camel Case Matches 118
Namespaces 118
Configuring Content Assist 119
Syntax Coloring 120
Automatic Completion 121
Matching Brackets 122
Mark Occurrences 123
Code Folding 125
Code Commenting 126
phpDoc Block Comments 127
Bookmarks 128
Hover Support 129

Configuring Hover Preferences 130
Override Indicators 131
PHP Working Sets 132
Type Hierarchy 133
PHP Manual Integration 134
Real Time Error Detection 136
Semantic Analysis 137
Quick Fix 137
Preferences 137
Local History 138
CVS 139
SVN 140
Zend Framework Development 141
Zend Framework Project and Element Creation 141
Zend Framework Content Assist 142
Database Connectivity 143
Running 144
PHP Script Local Running 144
PHP Script Remote Running 144
PHP Web Page Running 145
Debugging 146
PHP Script Local Debugging 146
PHP Script Remote Debugging 146
PHP Web Page Debugging 147
URL Debugging 147
Toolbar Debugging 148
Profiling 149
PHP Script Local Profiling 149
PHP Script Remote Profiling 149
PHP Web Page Profiling 150
URL Profiling 150
Toolbar Profiling 150
Profiler Views 151
Breakpoints 155
Conditional Breakpoints 155
PHP Include Paths 156
PHP Build Path 157

Path Mapping ... 158
Zend Browser Toolbar ... 161
Tunneling ... 162
Zend Server ... 164
Zend Server Integration ... 164
Viewing Events from Inside Zend Studio ... 165
Importing Events into Zend Studio ... 166
Code Tracing ... 167
PHPUnit Testing ... 168
Refactoring ... 170
JavaScript Support ... 172
PHPDocs ... 173
Code Galleries ... 174
Zend Guard Integration ... 175
RSS Feeds ... 176
WSDL - Web Services Description Language ... 177
Zend Studio for IBM i Extras ... 178
Content Assist ... 181
Remote System Explorer ... 182
Remote Server Support ... 183
Mylyn Integration ... 185
Phar Integration ... 186
Tasks ... 187
Tasks ... 187
Creating PHP Projects ... 188
Creating PHP Files ... 190
Creating a PHP File within a Project ... 190
Creating a PHP File Outside of a Project ... 190
Opening an External File ... 191
Creating PHP Elements ... 194
Creating a New PHP Class ... 194
Creating a New PHP Interface ... 196
Migrating From Zend Studio 5.X ... 198
Migrating Projects From Zend Studio 5.X ... 198
Migrating Keymaps from Zend Studio 5.X ... 200
Zend Server Integration ... 201
Zend Server Integration ... 201

Working with Code Tracing 203
Working with Code Tracing 203
Importing a Zend Server Event File 204
Opening the Trace Data Source 207
Debugging and Profiling Zend Server Events 209
Importing .xml Events Into Zend Studio 209
Debugging / Profiling Events from Zend Server 213
Setting Up Zend Server Integration 215
Configuring Zend Server Settings in Studio 215
Configuring Zend Studio in Zend Server 219
Configuring Studio Communication Settings in Zend Server 221
Configuring Zend Server to Auto Detect Studio Settings 222
Using Content Assist 223
Using Templates 224
Inserting a Template into Code 224
Formatting Code 226
Using Code Folding 227
Searching for PHP Elements 229
Hotkeys 231
Opening PHP Elements 232
Opening Types/Methods 233
Using Smart Goto Source 234
Viewing Type Hierarchies 235
Creating PHP Working Sets 237
Using Mark Occurrences 239
Finding and Replacing 240
Applying Quick Fixes 241
Adding Comments 242
Adding PHP DocBlock Comments 244
Using Local History 245
Comparing Files 246
Replacing Files 247
Restoring Deleted Files 247
Using CVS 248
Configuring a CVS Connection 249
Importing Projects from CVS 251
Accessing an Existing CVS Checkout 253

Uploading Projects to CVS 255
Using SVN 256
Configuring an SVN Connection 257
Importing Projects from SVN 259
Accessing an Existing SVN Checkout 261
Uploading Projects to SVN 263
Using FTP and SSH 264
Creating an FTP/SSH Connection 265
Editing the Default Port of an FTP/SSH Connection 266
Viewing and Editing Files on an FTP/SSH Server 267
Viewing and Editing Remote Server Files in your Workspace 268
Developing with Zend Framework 271
Creating Zend Framework Projects 272
Running and Debugging Zend Framework Projects 274
Setting Up Your Server Environment 274
Creating and Running a Zend Framework Example Project 278
Creating the Zend Framework Example Project 279
Running the Zend Framework Example Project 280
Creating Zend Framework Elements 281
Creating a Zend Module File 282
Creating a Zend Controller File 284
Creating a Zend Table File 286
Creating a Zend View File 287
Creating a Zend View Helper File 288
Creating a Zend Action Helper 290
Searching the Zend Framework Site 293
Using the Zend Tool Floating Window 294
Connecting to Databases 296
Connecting to Databases 296
Creating a Database Connection Profile 297
Connecting to a Database 300
Viewing and Editing Table Content 302
Creating a MYSQL Driver Definition 303
Creating and Executing an SQL Query 306
Running Files and Applications 308
Running PHP Scripts Locally 309
Running PHP Scripts Remotely 310

Running PHP Web Pages ... 312
Debugging Files and Applications ... 315
Setting Breakpoints ... 316
Debugging a PHP Script ... 318
Locally Debugging a PHP Script ... 319
Remotely Debugging a PHP Script ... 321
Debugging a PHP Web Page ... 323
Debugging a URL ... 326
Installing and Configuring the Zend Browser Toolbar ... 329
Running and Analyzing Debugger Results ... 334
Setting Up Remote Debugging ... 336
Setting your Environment to be an Allowed Host ... 337
Ensuring the Placement of dummy.php ... 339
Adding a Server Location Path Map ... 340
Setting Up Tunneling ... 342
Profiling Files and Applications ... 348
Profiling a PHP Script ... 349
Locally Profiling a PHP Script ... 350
Remotely Profiling a PHP Script ... 352
Profiling a PHP Web Page ... 354
Profiling a URL ... 356
Profiling Using the Zend Browser Toolbar ... 357
Managing PHP Libraries ... 358
PHP Libraries Preferences ... 358
Adding a PHP Library ... 360
Adding External Folders to PHP Libraries ... 361
Importing PHP User Libraries ... 364
Exporting PHP User Libraries ... 365
Editing PHP Library Components or Folders ... 366
Editing PHP User Libraries ... 367
Removing a PHP Library or Library Folder ... 368
Configuring a Project's PHP Include Path ... 369
Configuring a Project's PHP Build Path ... 371
Configuring Inclusion/Exclusion Patterns for the Project ... 372
Configuring Different Inclusion/Exclusion Patterns for Folders Within your Project ... 373
Adding External Source Folders to the Build Path ... 375
Adding a Server Location Path Map ... 377

Using PHPUnit Testing 379
Creating a PHPUnit Test Case 380
Running a PHPUnit Test Case 382
Creating a PHPUnit Test Suite 386
Running a PHPUnit Test Suite 387
Reporting on PHPUnit Test Results 388
Using Refactoring 390
Renaming Files 391
Renaming Elements 393
Moving Files 397
Extracting Variables 399
Extracting Methods 401
Generating Getters and Setters 402
Overriding / Implementing Methods 405
Creating a PHPDoc 407
Creating HTML files 410
Using Code Galleries 411
Inserting Code Snippets into your Script 412
Creating and Editing Code Gallery Entries 413
Interacting with Code Gallery Sites 415
Configuring Studio Communication Settings in Zend Server 418
Defining Zend Server in Studio 420
Integrating with Zend Guard 421
Encoding Projects Using Zend Guard 422
Opening and Editing Zend Guard Projects in Zend Studio 425
Working with WSDL 426
Incorporating WSDL Files 427
Viewing RSS Feeds 429
Working with Remote Server 430
Working with Remote Server 430
Creating a Connection via the Server View 431
Creating a New Remote Project 433
Downloading a Project from an Existing Connection 436
Sharing a Local Project to a Connection 437
Synchronizing a Local Project to a Connection 438
Upload to a Server 439
Download from a Server 439

Downloading Selected Files from a Folder 440
Uploading Resources to Your Server 441
Uploading a Project to a Remote Server 441
Importing Actions 442
Working with Mylyn Integration 443
Task List Presentation 443
Icon Legend and Color Coding 443
Creating new Tasks 443
Local Tasks 444
Repository Tasks 445
Context 445
Elements 446
Actions 446
Planning 446
Task-Focused Interface 447
Task-focused Ordering 448
Task Hyperlinking 448
Developing with JavaScript 449
Enabling JavaScript Support in PHP Projects 450
Enabling JavaScript Support for New PHP Projects 450
Enabling JavaScript Support for Existing PHP Projects 451
Removing JavaScript Support 451
Setting the JavaScript Build Path 452
Viewing JavaScript Elements in the Outline View 461
Using JavaScript Content Assist 462
Using JavaScript Syntax Coloring 465
Using JavaScript Mark Occurrences 467
Opening JavaScript Types 469
Setting Up and Using Dojo Integration 470
Reference 474
PHP Perspectives and Views 475
PHP Perspective 476
PHP Explorer View 477
Outline View 480
Remote Systems View 483
Servers View 486
Type Hierarchy View 488

PHP Debug Perspective ... 491
Debug View [PHP Debug Perspective] ... 493
Variables View [PHP Debug Perspective] ... 494
Breakpoints View [PHP Debug Perspective] ... 496
Parameter Stack View [PHP Debug Perspective] ... 498
Expressions View [PHP Debug Perspective] ... 499
PHP Profile Perspective ... 501
Profiling Monitor View ... 502
Profiler Information View ... 503
Execution Statistics View ... 504
Execution Flow View ... 506
Code Coverage Summary View ... 508
Zend Server Code Tracer Perspective ... 509
Tracer View ... 510
Additional Views ... 511
PHP Functions View ... 511
PHP Project Outline View ... 513
PHP Perspective Menus ... 514
File Menu ... 516
New Submenu ... 519
Import Submenu ... 523
Export Submenu ... 525
Edit Menu ... 527
Source Menu ... 529
Refactor Menu ... 531
Navigate Menu ... 532
Search Menu ... 534
Project Menu ... 536
Run Menu ... 538
Navigation Submenu ... 540
Window Submenu ... 541
Help Menu ... 543
PHP Perspective Main Toolbar ... 545
PHP Preferences ... 550
PHP Preferences Page ... 552
Appearance Preferences ... 553
Code Coverage Preferences ... 554

Code Gallery Preferences 556
Code Refactor Preferences 558
Code Style Preferences 559
Code Templates Preferences 559
Formatter Preferences 562
Debug Preferences 571
Installed Debuggers 573
Step Filtering Preferences 575
Workbench Options Preferences 577
Editor Preferences 578
Content Assist Preferences 579
Code Folding Preferences 581
Hovers Preferences 582
Mark Occurrences Preferences 584
Save Actions Preferences 586
Syntax Coloring Preferences 587
Task Tags Preferences 589
Templates Preferences 591
Typing Preferences 594
Zend Framework Preferences 596
New Project Layout Preferences 598
PHP Executables Preferences 600
PHP Interpreter Preferences 602
PHP Libraries Preferences 604
PHP Manual Preferences 606
PHP Servers Preferences 607
Tracer Preferences 610
Editing the Tracer Listener Settings 610
PHPUnit Preferences 612
Semantic Analysis Preferences 613
Configuring Tunneling Debug Preferences 616
Zend Guard Preferences 617
PHP Project Properties 618
PHP Project Properties 618
Resource Properties 619
Builders Properties 621
Code Style Properties 623

Code Templates Properties 623
Formatter Properties 624
Semantic Analysis Properties 625
PHP Build Path Properties 627
Formatter Properties 628
PHP Debug Properties 629
PHP Include Path Properties 630
PHP Interpreter Properties 631
PHP Task Tags Properties 632
Project References Properties 633
Refactoring History 634
Run/Debug Settings Properties 635
Save Actions Properties 636
Task Tags Properties 637
Validation Properties 638
PHP Icons 639
PHP Elements 639
Zend Framework Elements 640
Other Icons 640
Keymap 641
Useful Links 642
Support 643
Zend Support Center 643
Zend Forums 643
Zend Support Knowledge Base 643
Online Documentation 643
Open a Support Ticket 644
Zend Newsletter 644
Zend Developer Zone 644
Feedback 644
Registering Your License 645
Index 647

What's New in Zend Studio 7.2

The following new features are available in Zend Studio 7.2:

- Code Tracing

The following new features are available in Zend Studio 7.1:

- Remote System Explorer
- Phar Integration
- Mylyn Integration

The following new features are available in Zend Studio 7.0:

- **PHP 5.3 Support:**
  - Ability to create a PHP project based on PHP 5.3.
  - Syntax Coloring and Validation for PHP 5.3 in the editor
  - Conent Assist for Namespaces
  - Content Assist for packages
  - Namespace grouping in PHP Explorer
- **Zend Framework Enhancements**:
  - Zend Framework 1.8 Support - including code generation through Zend_Tool
  - Zend Tool Floating Window
  - Updated Zend Framework Example Project
- **Enhanced Code Editing:**
  - Mark Occurrences
  - Override Indicators
  - Type Hierarchy
  - Open Type/Method
  - Extract Variable
  - Content Assist Enhancements - Camel Case matches, Magic Members support, Namespaces
  - PHP Build Path Configuration
  - In-place Refactoring

- Zend Server Integration
  - Auto-detection of local Zend Server installation
  - Easy Zend Server Event Debugging / Profiling

Mac OS X

If you are running your Zend Studio on Mac OS X be aware that some Menu items may be labeled differently. This difference in labeling does not change the procedure.

For a list of the alternate labels, see the table below.

Windows	**Mac OS X**
Help | About Zend Studio	Zend Studio | About Zend Studio
Window | Preferences	Zend Studio | Preferences

Getting Started

The Getting Started section provides information on how to quickly and easily get started with the most commonly used Zend Studio functionality.

Quick Start - This Quick Start describes how to easily locate and use commonly used features, such as creating projects and files, Debugging PHP code, using Source Control (CVS, SVN), Refactoring and more.

Getting Started with Zend Studio - This guides you through the common process of using Zend Studio to develop and run your PHP applications.

Basic Tutorials - These tutorials demonstrate common processes along with examples and practical information on how to implement the information in order to improve your coding environment.

Switching from Zend Studio 5.x to Zend Studio 7.x - A quick guide for Zend Studio 5.x users on how to access commonly used functionality in Zend Studio 7.x.

Quick Start

The following Quick Start page will help newcomers and (even) our veteran users familiarize themselves with this new version.

The Features covered in the Quick Start are:

Getting Started
Workbench
Workspaces
Creating a PHP Project
Creating a PHP File
PHP Debugging
Remote Server
Source Control
PHPUnit
Refactoring
Perspectives of Interest

Getting Started

When Zend Studio is first launched, a Welcome Page will open containing links to actions and tutorials to help you get started with Zend Studio, as well as information on Zend Studio's features and functionality.

To start using Zend Studio, close the Welcome Page by clicking the X icon in the Welcome tab in the top-left corner.

Workbench

The Workbench is a window that displays Perspectives, Views and menu bars through which different operations can be performed.

See the Workbench topic in the Workbench User Guide for more on using and customizing the Workbench.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Workspaces

Workspaces are the files system location where all your projects are created and stored.

You can use the default Workspace created by Zend Studio or from the menu bar go to **File | Switch Workspace | Other** to select a different Workspace.

Creating a PHP Project

A project is a group of files and resources which are displayed in a tree in the PHP Explorer view.

To create a new PHP project:

Go to the Menu Bar and select **File | New | PHP Project**.

-Or- In the PHP Explorer View, right-click and select **New | PHP Project**.

See Creating PHP Projects for more information.

Creating a PHP File

To create a new PHP file:

Go to the Menu Bar and select **File | New | PHP File**.

-Or- in PHP Explorer view, right-click the folder in which you want to create your file and select **New | PHP File**.

-Or- click the New PHP File icon on the toolbar . This creates a file outside of a project.

See Creating PHP Files for more information.

PHP Debugging

The Zend Debugger detects and diagnoses errors in PHP code situated on local or remote servers:

To debug a PHP script situated on your workspace:

1. Set breakpoints at the relevant locations in your script by double-clicking the Marker Bar (located at the left of the editor area) at the relevant line. A blue ball appears to indicate that a breakpoint is set.
2. From the main menu, select **Run | Open Debug Dialog**
   -Or- right-click the file in PHP Explorer view and select **Debug As | Open Debug Dialog**.
3. To create a new configuration, double-click the 'PHP Script' category.
4. Under the Debugger Location category, choose whether you want to debug locally using the internal debugger (PHP executable) or remotely, using your server's debugger (PHP Web Server.
5. Enter all other information and click **Apply** and **Debug**.

To debug a PHP Web page situated on a server:

1. From the main menu, select **Run | Open Debug Dialog** -or- right-click the file in PHP Explorer view and select **Debug As | Open Debug Dialog**.
2. To create a new configuration, double-click the 'PHP Web Page' category.
3. Enter the required information and click **Apply** and **Debug**.

Debugging Preferences can be configured from the Debug Preferences page, which can be accessed from **Window | Preferences | PHP | Debug**.
See Debugging for more information.

Remote Server Explorer (RSE)

RSE (Remote Server Explorer) provides transparent access to remote resources, including the upload and download of files.

1. To configure a new connection, open the Servers View, situated at the bottom of the page, click the Connections node and from the right-click menu select **New | New Connections.**
2. Use the New Connection dialog to choose the connection type configure your settings and save.

To learn more about working with RSE, see Working with Remote Server Explorer.

Source Control

Zend Studio includes a built-in component for CVS (Concurrent Versions System) and SVN (Subversion).
Before accessing a repository, make sure that a CVS or SVN server is already configured on the host machine.

To configure access to a repository through Zend Studio:

1. On the main menu go to **Window | Show View | Other**.
2. Select either CVS or SVN repositories.
3. From the CVS/SVN view toolbar, select the **Add CVS/SVN Repository** button.
4. Fill in the location and authentication details and press **Finish**.
   CVS/SVN functionality can then be accessed by right-clicking on or within the relevant file or project and selecting **Team**, **Compare with**, **Replace with**, or **Source**.

See CVS or SVN for more information.

PHPUnit

A PHPUnit is a testing framework to write and run tests on PHP code. A test file can be created for each class, function and file. PHPUnits allow PHP developers to incrementally build test suites to constantly review progress and detect unintended side effects.

To create and run a PHPUnit Test Case:

1. In PHP Explorer view, right-click the file you want to test and select **New | PHP Unit Test Case**.
2. Fill in the required information in the New PHPUnit Test Case dialog.
3. Click **Finish** to create your Test Case file.
4. Edit the test functions in your new PHPUnit Test Case file by writing appropriate tests for the relevant functions.
5. Run the PHPUnit Test by going to **Run | Run As | PHPUnit** from the Menu Bar -or- right-clicking the file in PHP Explorer view and selecting **Run As | PHPUnit**.

See PHPUnit Testing for more information.

Refactoring

Refactoring is the process of renaming or moving selected resources in a 'smart' way while maintaining all the relevant links between files and elements. Refactoring automatically makes all relevant changes to your code.

To move/rename a resource:

1. Select the required file in PHP Explorer view -or- select the required element in the editor window.
2. Right-click and select **Refactor | Move** (files only)/**Rename**.
3. Select the resource's new location/name.

During the refactoring process, a preview screen will display showing the changes made.
See Refactoring for more information.

Perspectives of Interest

To open a perspective go to Window | Open Perspective, select "Other" to view a full list of perspectives.

- **PHP (default)** - This is the perspective that will open by default in Zend Studio. It allows you to manage and create PHP projects and files.
- **PHP Debug** - Allows you to manage and track the debugging process.
- **CVS / SVN Repository Exploring** - Allows you to manage and view your source control.
- **Database Development** - Allows you to view and manage your database content. Zend Studio allows connection with several types of databases.
- **PHP Profile** - Allows you "profile" and analyze file running times.
- **Source Code Repositories** - Allows you to upload and synchronize files through FTP.

See PHP Perspectives and Views for more information.

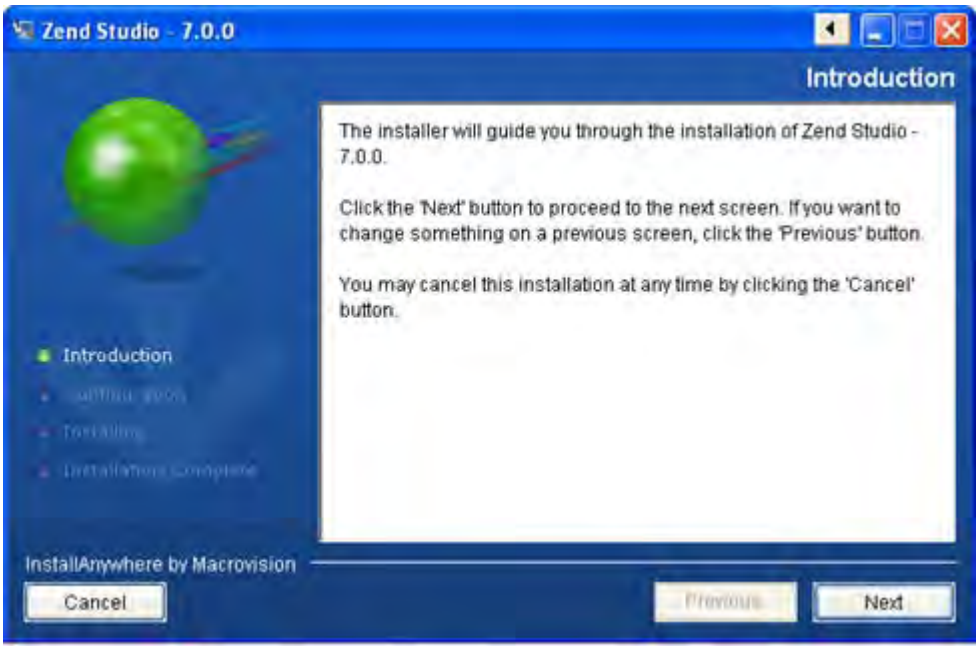
First Steps for Developing Applications with Zend Studio

Zend Studio, the leading development environment for PHP Web applications, maximizes developer productivity by allowing you to develop and maintain code faster, resolve application issues quickly and improve team collaboration. Backed by Zend and built on top the Eclipse platform, Zend Studio offers unmatched reliability and extensibility.

The following steps will help guide you through the basic process of using Zend Studio to develop and run your PHP applications:

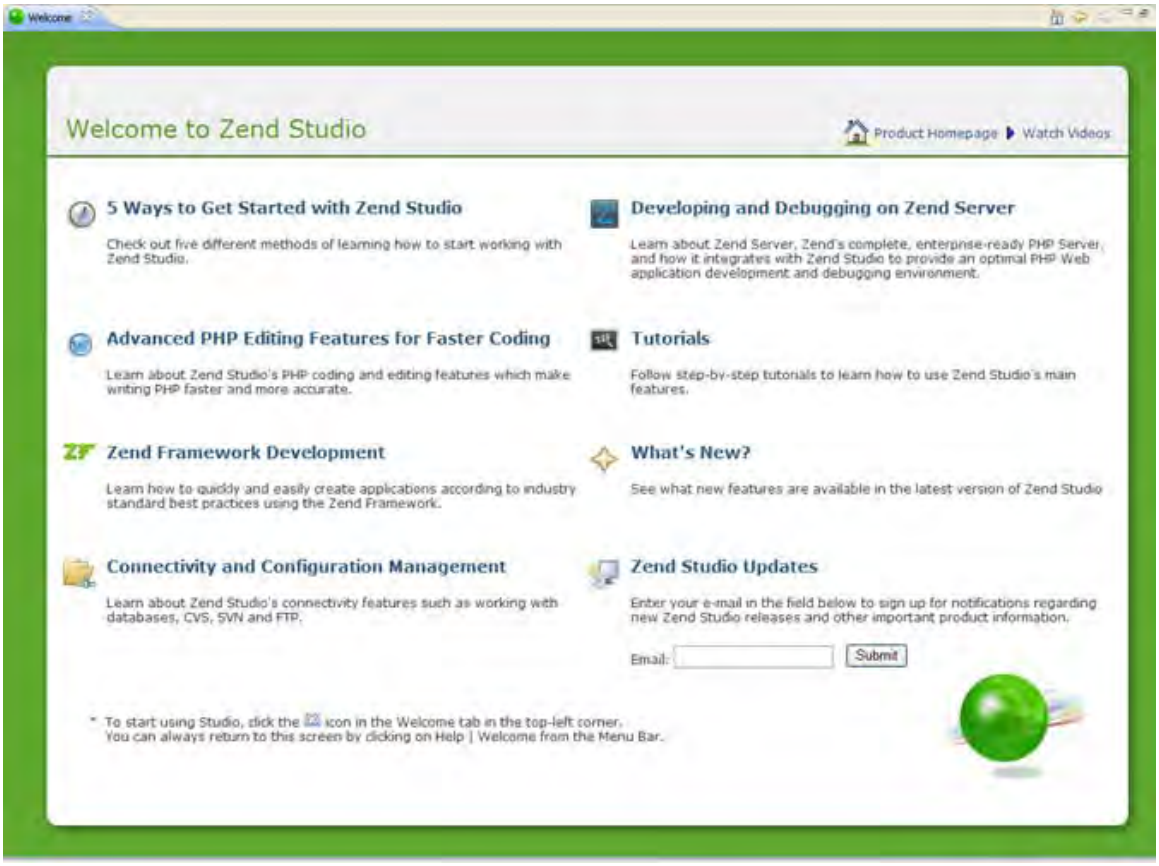
Install Zend Studio

Install Zend Studio by running the Zend Studio executable file and following the instructions in the dialogs.


Zend Studio - 7.0.0
Introduction
The installer will guide you through the installation of Zend Studio - 7.0.0.
Click the 'Next' button to proceed to the next screen. If you want to change something on a previous screen, click the 'Previous' button.
You may cancel this installation at any time by clicking the 'Cancel' button.
Introduction
InstallAnywhere by Macrovision
Cancel
Previous
Next

Launch Zend Studio

The first time Zend Studio is launched, the Welcome Page is displayed.


Welcome
Welcome to Zend Studio
Product Homepage
Watch Videos
5 Ways to Get Started with Zend Studio
Check out five different methods of learning how to start working with Zend Studio.
Developing and Debugging on Zend Server
Learn about Zend Server, Zend's complete, enterprise-ready PHP Server, and how it integrates with Zend Studio to provide an optimal PHP Web application development and debugging environment.
Advanced PHP Editing Features for Faster Coding
Learn about Zend Studio's PHP coding and editing features which make writing PHP faster and more accurate.
Tutorials
Follow step-by-step tutorials to learn how to use Zend Studio's main features.
Zend Framework Development
Learn how to quickly and easily create applications according to industry standard best practices using the Zend Framework.
What's New?
See what new features are available in the latest version of Zend Studio
Connectivity and Configuration Management
Learn about Zend Studio's connectivity features such as working with databases, CVS, SVN and FTP.
Zend Studio Updates
Enter your e-mail in the field below to sign up for notifications regarding new Zend Studio releases and other important product information.
Email:
Submit
To start using Studio, click the icon in the Welcome tab in the top-left corner.
You can always return to this screen by clicking on Help | Welcome from the Menu Bar.

To use the Welcome Page:

1. Browse the Welcome Page to learn about Zend Studio features and functionality.

   Click the home button in the top-right corner to be taken back to the main Welcome home page.
2. You can close the Welcome Page by clicking the icon in the Welcome tab, situated in the top-left corner of the window.Zend Studio's workbench is displayed, by default showing the PHP perspective. This perspective contains a number of views to assist you in PHP development.

   The PHP Explorer view is a file system view displaying the PHP projects located in your workspace.

Install Zend Server (optional)

Zend Server is a complete, enterprise-ready Web Application Server for running and managing PHP applications that require a high level of reliability, performance and security. It includes the most reliable and up-to-date version of PHP, tested PHP extensions, database drivers and other enhancements. Zend Server comes bundled with Zend Framework (the leading open-source PHP framework), Apache and MySQL.
Zend Server provides automatic integration with Zend Studio for an optimal environment for developing, deploying and debugging your PHP applications.

To install and configure Zend Server:

1. Download Zend Server from http://www.zend.com/en/products/server/downloads.

   Note:
   The Community Edition of Zend Server can also be downloaded from http://www.zend.com/products/server/downloads-all

2. Install Zend Server according to the installation instructions. These will vary depending on your operating system.
   See http://files.zend.com/help/Zend-Server/zend-server.htm#installation_guide.htm for full information on installing Zend Server.
3. If Zend Server is installed on the same machine as Zend Studio, Zend Studio will automatically detect and configure it.
   The auto detection can be triggered in two ways:
   i. Automatically when Zend Studio is launched.
   A popup balloon will appear in the bottom-right corner of the window indicating that a Zend Server installation has been detected and configured.

Zend Server Found
A Zend Server installation has been detected on your local machine and added as your default PHP server.
Click here to learn more

   ii. By clicking the Auto Detect Zend Server button in the Servers view (this is available from the default PHP Perspective or can be manually opened by going to **Window | Show View | Zend Servers | Servers**).

Once a Zend Server installation has been detected, the integration between Zend Studio and Zend Server enables you to easily deploy, run, debug and profile applications.

Create a PHP project

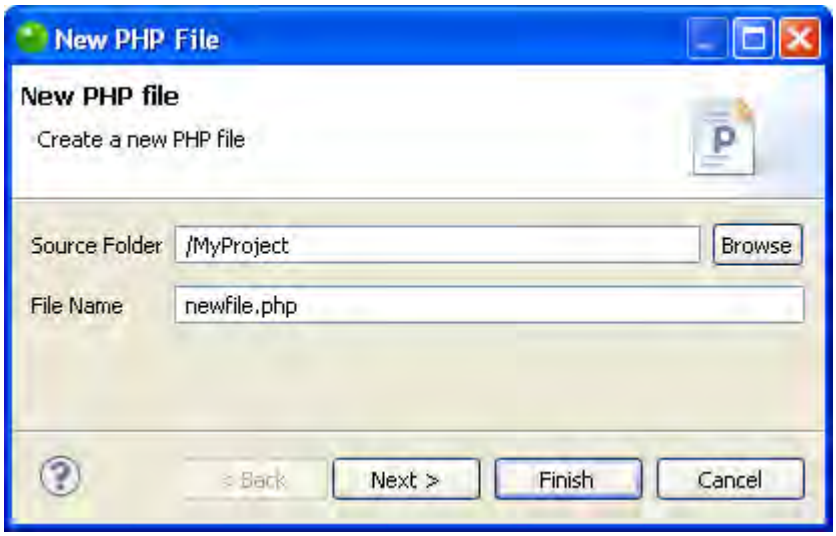
In order to start coding, you should create a PHP project which will contain the files for your application.

Create PHP Files

You can now start to develop your application by creating PHP Files in your project.

To create a new PHP file within you project:

1. In PHP Explorer view, select the project you created.
2. Right-click and select **New | PHP File** -or- go to File on the Menu Bar and select **New | PHP File**.
   The PHP File creation dialog is displayed.



3. Enter the name of the file and click **Finish**.

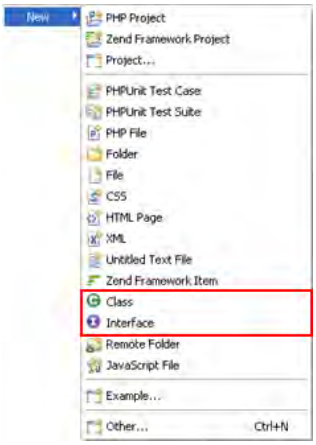
Your file will open in the editor and will appear within your project in PHP Explorer view.

Create PHP Elements

You can use Zend Studio's PHP element creation wizards to quickly and easily create PHP elements such as classes and interfaces within your code.

To create a new PHP class/interface

1. In PHP Explorer view, right-click the project/file in which you want to create the new class/interface and select **New | Class** or **Interface**.



The New PHP Class/Interface wizard is displayed

2. Enter the required details and click **Finish**.

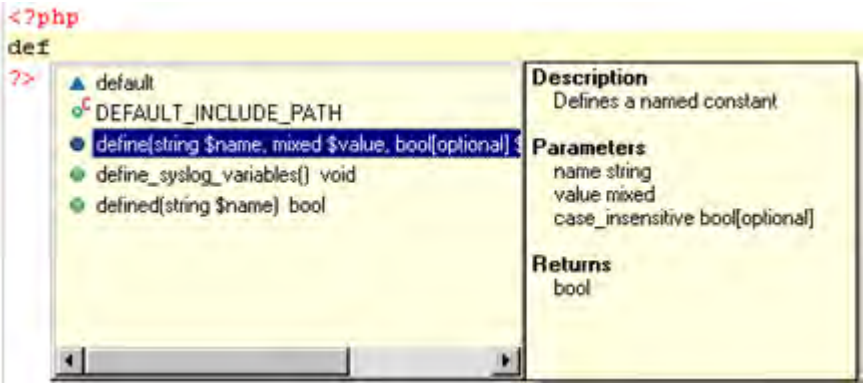
The new class/interface will be created with the required code.

Write Code

Manually type code in your files. As you type, content assist will suggest code options according to the context of the code.

You can select a required code string to quickly insert it into your script

If the Content Assist window does not pop up automatically, press **Ctrl+Space**.



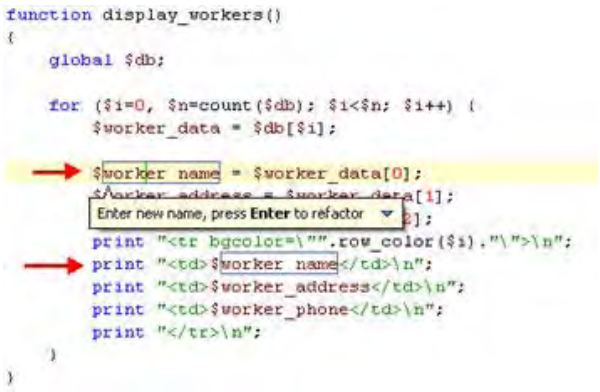
Rename Elements (Refactoring)

If at any point during your development you want to rename an element, this can be easily done using the in-place refactoring feature.
This will apply the rename operation to all occurrences of the required element so that all links between elements are maintained.

Note:
The refactoring feature also gives access to a range of other functions such as move file and extract variable. See the Refactoring topic in the Zend Studio Online Documentation or User Guide for more information.

To rename an element within the editor:

1. In the editor, place your cursor on the element to be renamed.
2. From the menu bar select **Refactor | Rename** -or- right-click and select **Refactor | Rename** -or- press **Alt-Shift-R**.
   All occurrences of the element are put in a frame and the Refactor popup is displayed.



4. Type the new element name in the box.
   All occurrences of the element name are automatically updated.

   Note:
   To preview the changes before applying them, click the arrow in the right-hand corner of the Refactor popup and select Preview.

5. Click **Enter** to apply the refactoring.

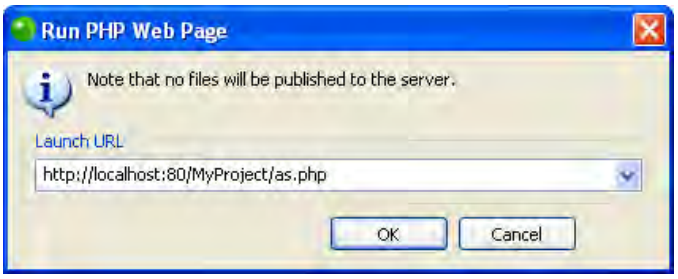
The element will be renamed and all instances where that element is referenced will be updated to reflect the changes.
See Refactoring for more information.

Run your application

In order to preview the execution of your application on a server, you can easily run it in Zend Studio's internal browser.

To run your application:

1. Place your application on a Web server.
   If you selected to create your project on a local Zend Server you can skip this step.
2. In PHP Explorer, right-click the file from which you would like to start the execution and select **Run As | PHP Web Page**.
   The Run PHP Web Page dialog opens.



3. Ensure the URL pointing to the file on the server is correct.
   Edit if necessary.
4. Click **OK**.

Your application will be run and displayed in a browser.

You can periodically rerun your application to preview any changes you have made to your files during development. The files that will be executed will be the files situated locally on your workspace, irrespective of any changes made to the server copies of the files.

Next Steps

In addition to the options and features highlighted in this document, Zend Studio contains a wide range of features to help you with all aspects of your PHP development.

To learn more about the full range of funcitonality offered by Zend Studio:

- See the Zend Studio Online Documentation or User Guide for full information on all the features available in Zend Studio.
- Visit the demo video library (http://www.zend.com/en/products/studio/videos) to view short demo videos on Zend Studio's capabilities.
- Visit the Zend Studio Support Center (www.zend.com/en/support-center) to get answers to frequently asked questions from the forums and knowledgebase articles.

Switching from Zend Studio 5.x to Zend Studio 7.x

The following 'Switching from Zend Studio 5.x to Zend Studio 7.x' quick guide' is intended for users who have worked with the traditional Zend Studio and want to learn how to perform Zend Studio 5.x tasks in the new Zend Studio 7.x.

Contents:

Project/File Creation
Debugging
Profiling
Source Control - Subversion (SVN)
Source Control - CVS
FTP Connectivity
Database Connection
Tunneling

Project/File Creation

To create a new PHP project:

Zend Studio 5.x	Zend Studio 7.x
1. From the main toolbar select Project | New Project. The New Project Wizard dialog box will appear. 2. Enter a name for the new project. The location is updated accordingly. 3. Click Next to define specific properties for the new project. 4. Click Finish.	1. Go to File Menu and select New | PHP Project -Or- In the PHP Explorer View, right-click and select New | PHP Project. 2. The New Project wizard will open. Enter a name for your new project into the Project Name field. 3. Click Finish.

See the Creating Projects and Files tutorial for more information.

To create a new file:

Zend Studio 5.x	Zend Studio 7.x
1. From the Menu Bar select File | New File -Or- Click the New File icon on the toolbar. 2. A new PHP file will open in the editor.	**To create a new PHP file not associated with a project:** 1. Click the new Easy PHP File icon on the toolbar -Or- In PHP Explorer view, right-click and select New | Untitled PHP Document. 2. A new PHP file, by default called PHPDocument1, will open in the editor. **To create a new PHP file within a project:** 1. In PHP Explorer view, select the Project within which you would like to place the file. 2. Right-click and select New | PHP File -or- go to File on the Menu Bar and select New | PHP File. 3. The PHP File creation dialog will be displayed. 4. Enter the name of the file and click Next. 5. Click Finish.

See Easy File Creation for more information.

Debugging

Note:

By creating a debug launch configuration in Zend Studio 7.x, you can easily rerun the debug session with the settings specified by clicking the arrow next to the debug button on the toolbar and selecting your launch configuration.

To debug a PHP script using Zend Studio's internal debugger:

Zend Studio 5.x	Zend Studio 7.x
1. Open the Preferences window by selecting Tools | Preferences from the main menu. 2. Select the Debug tab. 3. From the Debug Server Configuration area of the Debug tab, select 'internal" from the Debug Mode category. 4. Click Apply and OK. 5. In the main toolbar, click Go to start the Debugger. -or- from the Menu Bar select Debug | Go.	1. Click the arrow next to the debug button on the toolbar and select Debug As | PHP Script. -Or- In PHP Explorer view, right-click the required file and select Debug As | PHP Script.

See Locally Debugging a PHP Script for more information.

To debug a PHP script using your server debugger:

Zend Studio 5.x	Zend Studio 7.x
1. Open the Preferences window by selecting Tools | Preferences from the main menu. 2. Select the Debug tab. 3. From the Debug Server Configuration area of the Debug tab, select 'server' from the Debug Mode category. 4. Enter the URL of the server on which you want to Debug your files. 5. Click Apply and OK. 6. In Zend Studio's main toolbar, click Run to start the Debugger.	1. Click the arrow next to the debug button on the toolbar and select Debug Configurations... -or- In PHP Explorer view, right-click and select Debug as | Debug Configurations... A Debug dialog will open. 2. Double-click the PHP Script option to create a new debug configuration and enter a name for it. 3. Select the PHP Web Server option under the Debugger Location category and select your server from the list. 4. Under PHP File, click Browse and select your file. 5. Click Apply and then Debug. **Note:** The next time you want to run this debug session, click the arrow next to the debug button on the toolbar and select your launch configuration.

See Remotely Debugging a PHP Script for more information.

To debug applications on a server:

Zend Studio 5.x	Zend Studio 7.x
1. From the Menu Bar, select Debug | Debug URL. The Debug URL dialog will appear. 2. Enter the URL of the file/application which you would like to debug. 3. Select whether to use the local copy or server copies of the files. 4. Click OK.	1. Click the Debug URL button on the main toolbar -or- go to Run | Debug URL.. The Debug URL dialog will appear. 2. In the 'Open Browser at' field, enter the URL of the first page that should be debugged. 3. Click Debug. **Note:** In Zend Studio 7.x, creating a Debug launch configuration gives you access to advanced Debugging options, and allows you to easily re-run past Debug sessions using the same settings. **To create and execute a Debug Launch Configuration for debugging applications on a server:** 1. Click the arrow next to the debug button on the toolbar and select Debug Configurations... –or- select Run | Debug Configurations... A Debug Configurations dialog will open. 2. Double-click the PHP Web Page option to create a new debug configuration. 3. Enter a name for the new configuration. 4. Select your server from the PHP Server drop-down list. 5. Under PHP File, click Browse and select your file. 6. Select the Advanced tab. 7. In the 'Source Location' category, select whether to use the local copy or server

Zend Studio 5.x	Zend Studio 7.x
	copies of the files. 8. Click Apply and then Debug. **Note:** The next time you want to run this debug session, click the arrow next to the debug button on the toolbar and select your launch configuration.

See Debugging a PHP Web Page for more information.

Profiling

Note:
In addition to Profiling applications on a server, Zend Studio 7.x includes the option to profile PHP Scripts situated on your workspace using the internal debugger or your server debugger. See Profiling a PHP Script for more information.

To profile applications on a server:

Zend Studio 5.x	Zend Studio 7.x
1. From the Menu Bar, select Debug | Profile URL. The Profile URL dialog will appear. 2. Enter the URL of the file/application which you would like to Profile. 3. Select whether to use the local copy or server copies of the files. 4. Click OK.	1. Click the profile URL button on the main toolbar -or- from the Menu Bar, go to Run | Profile URL. The Profile URL dialog will appear. 2. In the 'Open Browser at:' field, enter the URL of the page that should be profiled. 3. Click Profile. **Note:** In Zend Studio 7.x, creating a Profiling launch configuration gives you access to advanced Profiling options, and allows you to easily re-run past profile sessions using the same settings. **To create and execute a Profile Launch Configuration for profiling applications on a server:** 1. Click the arrow next to the Profile button on the toolbar and select Profile Configurations &endash;or- select Run | Profile Configurations. A Profile dialog will open. 2. Double-click the PHP Web Page option to create a new Profile configuration. 3. Enter a name for the new configuration. 4. Select your server from the PHP Server drop-down list. 5. Under PHP File, click Browse and select your file. 6. Select the Advanced tab. 7. In the 'Source Location' category, select whether to use the local copy or server copies of the files. 8. Click Apply and then Profile.

Zend Studio 5.x	Zend Studio 7.x
	Note: The next time you want to run this profile session, click the arrow next to the profile button on the toolbar and select your launch configuration.

See Profiling a PHP Web Page for more information.

Source Control - Subversion (SVN)

See the Subversive User Guide for more information on SVN.

Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

To define Subversion connectivity:

Zend Studio 5.x	Zend Studio 7.x
1. Open the Preferences window by selecting Tools | Preferences from the Menu Bar. 2. Select the Source Control tab. 3. Select 'Subversion' in the Source Control drop-down list. This will cause all source control preferences and menu options to enable Subversion rather than CVS operations. 4. Configure any required Subversion settings. 5. Click Apply and OK.	1. Open the SVN perspective by going to Window | Open Perspective | Other | SVN Repository Exploring. 2. In the SVN Repositories view, click the Add SVN Repository button on the view's toolbar -or- right-click within the SVN view and select New | Repository Location. The Add SVN Repository dialog will open. 3. Enter the information required to identify and connect to your repository. 4. Click Finish. Your SVN repository will be added to the SVN Repository view. **Note:** The connection details will be saved and can be easily selected when performing SVN actions on your projects and files.

See Configuring an SVN Connection for more information.

Checking out a Module from SVN

Zend Studio 5.x	Zend Studio 7.x
1. Set Subversion to be your Source Control default by following the steps under 'To define Subversion connectivity', above. 2. Go to Tools | Subversion | Checkout. 3. Enter the repository details. 3. Set the Checkout Options. 4. Click OK.	1. Create an SVN repository connection by following the steps under 'To define Subversion connectivity', above. 2. Go to File | Import | Projects from SVN and click Next. 3. Select your repository and click Next. 4. Select the 'Use the Repository URL as the label' option and select the required module/project to check out. 5. Click Finish.

See Importing Projects From SVN for more information.

To perform Subversion commands (e.g. update/commit) on an SVN module:

Zend Studio 5.x	Zend Studio 7.x
1. Set Subversion to be your Source Control default by following the instructions under 'To define Subversion connectivity', above. 2. Right-click the required file/project in the Project tab -or- open the file in an editor and right-click. 3. From the right-click menu, select Subversion and the required action.	1. Create an SVN repository configuration by following the steps under 'To define Subversion connectivity', above. 2. Right-click the required file/project in PHP Explorer -or- open the file in an editor and right-click. 3. From the right-click menu, select Team and the required action.

See the "Making Changes, Comparing Changes, and Committing Changes" section of the Working with SVN Tutorial for more information.

See 'Working in a Team Environment with CVS' in the Workbench User Guide for more information.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

To define CVS connectivity:

Zend Studio 5.x	Zend Studio 7.x
1. Open the Preferences window by selecting Tools | Preferences from the Menu Bar. 2. Select the Source Control tab. 3. Select 'CVS' in the Source Control drop-down list. This will cause all source control preferences and menu options to enable CVS rather than Subversion operations. 4. Configure any required CVS settings. 5. Click Apply and OK.	1. From the Menu Bar, open the CVS perspective by selecting Window | Open Perspective | Other | CVS Repository Exploring from the Menu Bar. 2. Click the Add CVS Repository button on the view's toolbar -or- right-click within the CVS view and select New | Repository Location. The Add CVS Repository dialog will open. 3. Enter the information required to identify and connect to the repository location. 4. Click Finish to create your connection. **Note:** The connection details will be saved and can be easily selected when performing CVS actions on your projects and files.

See Configuring a CVS Connection for more information.

Checking out a Module from CVS

Zend Studio 5.x	Zend Studio 7.x
1. Set CVS to be your Source Control default by following the instructions under 'To define CVS connectivity', above. 2. From the Menu Bar, go to Tools | CVS | Checkout. 3. Enter the details in the Checkout dialog. 4. Click OK.	1. Go to File Menu and select Import | Projects from CVS and click Next. 2. Select your repository and click Next. A 'Select Resource' dialog will appear. 3. Select your project (if necessary, expand the nodes until you see it) and click Finish. A 'Check Out As' dialog will appear. 4. Click Finish.

See Importing Projects from CVS for more information.

To perform CVS commands (e.g. update/commit) on a CVS module:

Zend Studio 5.x	Zend Studio 7.x
1. Set CVS to be your Source Control default by following the instructions under 'To define CVS connectivity', above. 2. Right-click the required file/project in the Project tab -or- open the file in an editor and right-click. 3. From the right-click menu, select CVS and the required action.	1. Right-click the required file/project in PHP Explorer View -or- open the file in an editor and right-click. 2. From the right-click menu, select Team and the required action.

See the "Making Changes, Comparing Changes, and Committing Changes" section of the Working with CVS tutorial for more information.

FTP Connectivity

To configure an FTP root:

Zend Studio 5.x	Zend Studio 7.x
1. Go to the main menu and select File | Add FTP Server -Or- from the File Manager's File System tab, open the right-click menu and select Add FTP Server. 2. The Configure FTP Server dialog will appear. 3. Enter your FTP connection details and click OK. The new FTP Icon appears in the file system.	1. Switch to the Remote Systems view (tabbed with the PHP Explorer view) and click the 'Define a connection to remote system' button on the view's toolbar. The New Connection dialog will appear. 2. Select 'FTP' and click Next. 3. Enter your Remote FTP connection details and click Finish. Your connection will created and listed in the Remote Systems view.

See Creating an FTP/SSH Connection for more information.

Database Connection

Note:
Zend Studio 7.x allows connection to a variety of database types.

To create an SQL server connection:

Zend Studio 5.x	Zend Studio 7.x
1. From Studio's File Manager, click the SQL tab . 2. Right-click and select Add Server. The Add SQL Server dialog will open. 3. Enter the required SQL Server Settings. 4. Click OK. Your Server Tree will be added to the SQL tab.	1. In Database Development Perspective, click the Create New SQL Connection icon on the Data Source Explorer view toolbar. The New JDBC Connection Profile wizard opens. 2. Select a driver from the list and enter the required information. 3. Click Finish. Your new connection profile will be added to your databases list in the Data Source Explorer view.

See Creating a Database Connection Profile for more information.

To connect to your SQL database server:

Zend Studio 5.x	Zend Studio 7.x
1. Create an SQL server connection by following the instructions under 'To create an SQL server connection', above. 2. Select the SQL tab from Studio's File Manager. 3. Double-click the Server you want to connect to. The SQL Database tree will be displayed.	1. Create an SQL server connection by following the instructions under 'To create an SQL server connection', above. 2. Open the Database Development perspective by going to Window | Open Perspective | Other | Database Development. 3. In the Data Source Explorer view, expand the SQL Databases node, right-click your SQL server connection profile and select Connect. The SQL Database tree will be displayed.

See Connecting to a Database for more information.

To view the contents of a database table:

Zend Studio 5.x	Zend Studio 7.x
1. Connect to your SQL database server by following the steps under 'To connect to your SQL database server', above. 2. Expand the server tree until you see the required table. 3. Double-click the table -or- right-click the table and select Show Table Data. 4. In the Results in Page selector at the top of the Data Display, select the number of results you wish to display per screen. The Data Display will show the contents of the table according to the selected resolution.	1. Connect to your SQL database server by following the steps under 'To connect to your SQL database server', above. 2. Expand the server tree until you see the required table. 3. Double-click the required table or right-click it and select Data | Edit. The table will open in a database editor displaying all the data within the table.

See Viewing and Editing Database Table Content for more information.

Tunneling

Note:

Zend Studio 7.x allows you to define tunneling connections to a number of servers, which can then be easily selected to establish a connection.

To select a server to connect to using a tunneling connection, define the server following the instructions below, click the arrow next to the tunneling icon on the toolbar and select the required server from the list.

To configure tunneling preferences in Zend Studio:

Zend Studio 5.x	Zend Studio 7.x
1. Go to Tools | Tunneling Settings. 2. Define the relevant tunneling settings. 3. Click Connect.	1. Open the PHP Server Preferences page by going to the Menu Bar and selecting Window | Preferences | PHP | PHP Servers. 2. Click **New** to define a New Server (or **Edit** if the server has already been defined). 3. Enter any relevant settings in the Server, Path Mapping and Zend Server Integration tabs. 4. In the Tunneling Settings tab, check the "**Enable Tunneling"** option and enter all necessary information. 5. Click **Finish** or **OK**. You can now use this server configuration to connect to your server using a tunneling connection by clicking the Tunneling icon on the toolbar .

See Setting Up a Tunneling Server for more information.

Note:

To configure a tunneling connection between Zend Studio and your server, settings also need to be configured on your server.

See Setting Up Tunneling for full instructions on setting up a Tunneling connection with your server.

Basic Tutorials

The Basic Tutorials section contains short tutorials on popular tasks that can be performed with Zend Studio. Each tutorial covers workflow issues from A-Z describing the processes and workflow that should be followed in order to complete the tasks.

Get up and running with one of these tutorials:

Creating Projects and Files
Working with Content Assist
Working with the Debugger
Working with CVS
Working with SVN
Refactoring
Working with PHPUnit testing
Working with the Profiler
Working with Zend Server

Creating Projects and Files

The purpose of this tutorial is to guide you through the steps involved in creating PHP Projects and files.

Contents:

Creating a PHP Project

Creating a PHP File

Creating a PHP Project

To create a new PHP Project:

1. Go to File Menu and select **New | PHP Project**.
   -Or- In PHP Explorer view, right-click and select **New | PHP Project**.
2. The New Project wizard will open.
   Enter a name for your new project into the Project Name field.
3. Click **Finish** to complete the creation of your project.

The new project will be listed in PHP Explorer view.

Creating a PHP File

Creating a PHP file within Zend Studio will automatically add PHP tags to the script, and allow you to fully utilize Zend Studio's PHP functionality.

To create a new PHP file within a project:

1. In PHP Explorer view, right-click your project and select **New | PHP File** -or- select your project and go to **File Menu | New | PHP File**.
2. Enter the File Name and click **Finish**.
   An editor window will appear with the following basic PHP code:

```
1 <?php
2
3 ?>
```

3. Add your code to the new file.
4. Save the file by clicking the Save button on the toolbar.
5. To complete your project, create more PHP files by repeating steps 1-7.

To create a new PHP file outside of a project:

1. Click the new Easy PHP File icon on the toolbar .
   An editor window will appear with the following basic PHP code:

```
1 <?php
2
3 ?>
```

2. Add your code to the new file.
3. Save the file by clicking the Save button on the toolbar.
4. To complete your project, create more PHP files by repeating steps 1-7.

Once you have created all your files, you can edit, debug, profile, test and monitor them. After taking all the necessary steps in order to obtain a server, you will be ready to deploy the project to the live server on which it will be running.

For more information on methods of deploying and synchronizing information with a remote server see: Working with Remote Server.

Working with Content Assist

The purpose of this tutorial is to teach you how to use Zend Studio's Content Assist function in order to write code quickly, easily, and accurately.

Contents:

Purpose and Usage
Activating Content Assist
Making Content Assist Elements Available Within the Same Scope
Function Parameter Hints
Class Type Hints
Configuring Content Assist

Purpose and Usage

The Content Assist feature enables the selection and insertion of existing code elements to complete partially entered code.
A list of possible code elements appears in relevant locations within your files according to the context of your cursor, which you can then select to be automatically entered into your code.

Each type of code element will have a unique icon:

Reserved PHP Words

Functions

Templates

Classes

Interfaces

Constants

 Variables (public)

Content Assist works with the following elements: PHP Classes, Functions, Variables, Constants, Keywords, Interfaces, Attributes, Values, Nested Functions, Names and Syntax, as well as all user defined Classes, Functions, and Constants.

Note:
Content Assist works with both PHP and HTML.

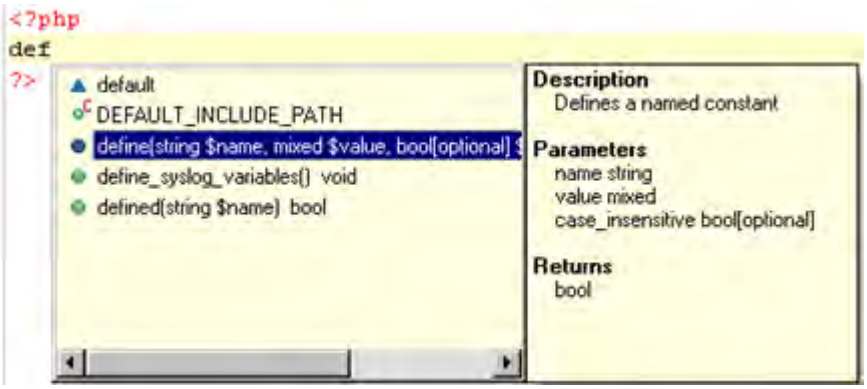
Activating Content Assist

By default, the Content Assist options will be automatically displayed once the first few characters of the code have been entered.

The following procedure demonstrates using Content Assist:

1. Create a new PHP File called 'File1'.
2. On the line beneath the opening PHP tag, type "def".
3. Click **Ctrl+Space**.

   The Content Assist window will be displayed with a list of suitable code completion options:



4. Double-click the first define function from the Code Completion window -or- select it and press **Enter**.

"define()" appears on the edit line.

Note:

If the Content Assist window does not open automatically, place your cursor at the required location and press **Ctrl+Space**.

To enable the Content Assist window to open automatically, go to the Content Assist Preferences page, accessed from **Window | Preferences | PHP | Editor | Content Assist** and mark the 'Enable auto-activation' checkbox.

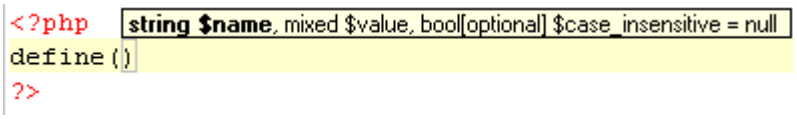
Function Parameter Hints

When entering a function call, a Function Parameter Hint box will be displayed detailing the types of parameters that should be entered within the parentheses of the function call.

The following procedure demonstrates using the Function Parameter Hint feature:

1. Place your cursor between the parentheses of the above function call: "define()"
2. Press **Ctrl+Shift+Space**.

A function parameter hint box will be displayed indicating the types of parameters that should be inserted between the parentheses.



Making Content Assist Elements Available Within the Same Scope

Added Code - Available within the same function and file

Elements within the same scope (e.g. within the same project, file or function) will be available for use with Content Assist.
Variables belonging to different scopes are not available to each other via Content Assist.

The following procedure demonstrates using Content Assist for inserting elements within the same function and file:

1. Edit your PHP File ('File1') so that it contains the following code:

```
<?php
define('continent','africa');
$control = '';
$mail = 'int@eclipse.org';
function media() {
$music = '';
$messenger = '';
$                        /*--------- Location_1*/
}
$                                  /* ---------Location_2*/
?>
```

2. Place the cursor at the "$" marked by "Location_1". This is within function "media".

3. Type the letter "m". The Content Assist window will be displayed with the variables "$messenger" and "$music", which were defined within the function.
   Note that the variable $mail (not within the scope of "media()") is not available.
4. Next, place the cursor at the "$" marked by "Location_2".
5. Type the letter "m". The Content Assist window will be displayed with the variable $mail, which is within the same file.
   Note that media's variables - $music and $messenger - are not within the function 'media' and so are not displayed.
6. Select 'mail' from the Content Assist window to insert it into the script.

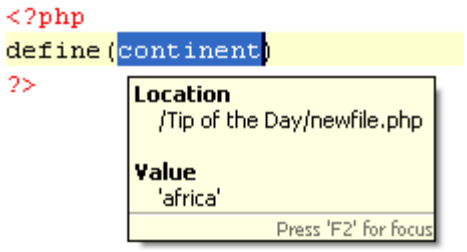
Added Code - Available Within the Same Project

Code elements defined in one file are also available for use by other files within the same project.

The following steps demonstrate using Content Assist for inserting elements within the same project:

1. Within the same project as "File1", create a new PHP file called "File2".
2. On the line beneath the opening PHP tag, type "def" and press **Ctrl+Space** to activate Content Assist. Double-click one of the define options.
3. Between the parentheses, type "cont" and press **Ctrl+Space** to activate Content Assist. The element 'continent', defined in "File1", will be available.
4. Double-click 'continent' to enter it into your code.

When the element is highlighted, Content Assist displays the original location of the code element, its value ('africa') and all other information available.



Class Type Hints

By using a comment you can assign a variable its exact class value. This assignment will affect the content assist of this variable accordingly.

To see and trial this feature:

1. Create a new PHP file with the following code:

```
<?php
function getClass() {
return new Test();
}
class Test {
function printValues($a, $b) {
echo "Values: $a, $b";
  }
}

/* @var $myVar Test */
$myVar = getClass();

$myVar->
?>
```

2. Place your cursor after '$myVar->' (on the line above the closing PHP tag) and press Ctrl+Space to activate Content Assist. Content Assist will open with the function defined in 'Test' class (printValues($a, $b)). Double click it to enter it into your code.

Note:

Without the comment, Content Assist will not be available for the function.

Configuring Content Assist

Content Assist options can be configured through the Content Assist Preferences page, accessible from **Window | Preferences | PHP | Editor | Content Assist**.

Working with CVS

The purpose of this tutorial is to teach you how to work with the CVS source control system. You will learn how to configure your CVS repository, upload projects and files to it, check out (import) projects and files from it and commit changes which you have made to files.

Contents:

Purpose
Adding a CVS Repository
Sharing Projects
Checking Out Projects from CVS
Adding Files to Existing Projects
Making Changes, Comparing Changes and Committing Changes
Replacing Files with Older Versions
Deleting Files from CVS

Purpose

A Concurrent Versions System (CVS) repository is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

Adding a CVS Repository

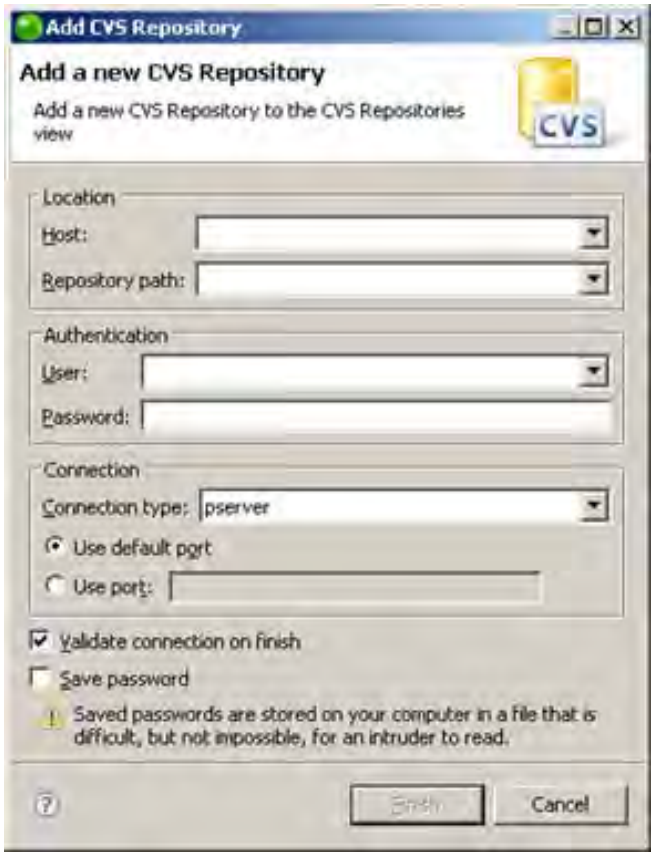
Before you can add projects to or export projects from CVS, you must define your CVS repository settings.

Note:
To access a repository, make sure that a CVS server is already configured.

This procedure describes how to create a CVS repository connection which you can access in order to be able to carry out CVS functionality.

To create a new CVS repository connection:

1. Open the CVS perspective by going to Window menu and selecting **Open Perspective | Other | CVS Repository Exploring**.
2. Click the Add CVS Repository button on the view's toolbar -or- right-click within the CVS view and select **New | Repository Location**.
   The Add CVS Repository dialog will open.



3. Enter the information required to identify and connect to the repository location:
   - Host - The host address (e.g. mycomputer.com).
   - Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
   - User - The user name with which you connect to the repository.
   - Password - The password for the user name.
   - Connection Type - The authentication protocol for the CVS server.
     There are four connection methods:
     i. pserver - a CVS specific connection method.
     ii. extssh - an SSH 2.0 client.
     iii. pserverssh2 - provides a pserver connection over ssh2.
     iv. ext - the CVS ext connection method which uses an external tool such as SSH to connect to the repository.
   - If the host uses a custom port, enable Use Port and enter the port number.
4. Click Finish to create your connection.

Your repository will be displayed in the CVS Repositories view.

Sharing Projects

Through CVS, projects can be shared and worked on by numerous team members.

This procedure demonstrates how to upload a project you have created so that other users can work on it:

1. Create a new PHP project called "MyCVS Project".
2. Within the project, create a PHP file called "CVSFile1" with the following code:
   ```
   <?php
   //This is a new file
   ?>
   ```
3. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A Share Project dialog will open.
4. From the repository list, select CVS and click **Next**.
5. Select 'Use existing repository location', and select your CVS repository from the list.
6. Click **Next**.
7. In the Module Name dialog, select 'Use project name as module name'.
8. Click **Next**.
9. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click **Next**. (Mark the Save Password checkbox to

ensure that this screen does not reappear.)

10. The 'Share Project Resources' dialog will open.
    Your project will be displayed as a resource that is to be added to CVS. The purple plus icon indicates that these are new files that have not previously been added to CVS.
11. Click **Finish**.
    A Commit dialog will open.
12. Enter the comment "I am uploading files to CVS." and click **OK**.
13. In PHP Explorer View, your project will now have a repository icon , indicating that it is in CVS.
14. Once you have committed your files, other team members will be able to access and change the files.

The instructions below explain how users can check out (import) projects from CVS, edit them and upload their changes.

Checking Out Projects from CVS

Once projects are placed on the CVS repository, they can be checked out (imported) by anyone with access to that repository.

This procedure demonstrates how to import (check out) projects from CVS into your workspace:

1. Delete the 'MyCVS Project' from your workspace in order to simulate being a new user who has not previously had access to this file.
   Note: Deleting the project from your workspace will not delete it from CVS.
2. Go to **File | Import | Projects from CVS**.
3. Click **Next**.
4. Select your repository.
5. Click **Next**.
6. Select the 'Use an existing module' option to see all the available projects under that directory.
7. Select the 'MyCVS Project'.
8. Click **Finish**.

The project will now be added to your workspace and will be displayed in the PHP Explorer view. Note the repository icon next to the project in PHP Explorer view, indicating that they it is sitting in a CVS repository.

Adding Files to Existing Projects

You can add files to existing projects in the CVS repository and commit them so that other users can access them.

This procedure demonstrates how to add and commit a file into an existing project:

1. In your MyCVSProject, create a new PHP file, called "CVSFile2", with the following code:

```
<?php
//Another new file
?>
```

2. Save the file.
3. In PHP Explorer view, select the file, right-click and select **Team | Commit**.
4. A Commit dialog will open.
   Enter a comment "Another new file added." and click **OK**.
5. The file will be committed to CVS and will be accessible by other users.

Note the icon next to the file, indicating that it is sitting in a CVS repository.

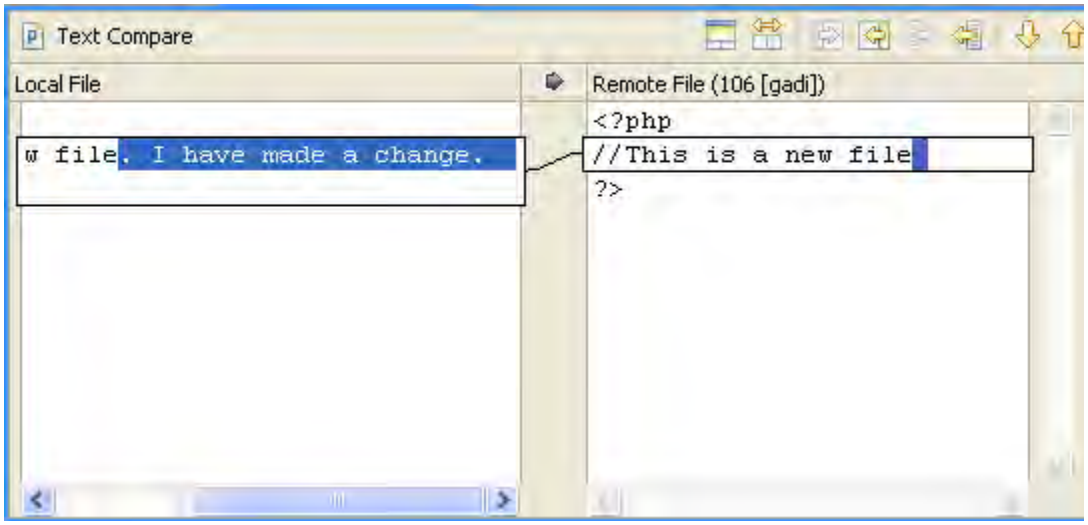
Making Changes, Comparing Changes, and Committing Changes

Once files are stored on CVS, you and all other team members can make changes to the files and commit them. Before committing changes you have made to a file, you can compare the file stored locally in your workspace to the file stored on the CVS repository.

Making and Comparing Changes

This procedure demonstrates how to make changes to files and compare the newly edited local files to files in the repository:

1. Open CVSFile1 in your MyCVS Project.
2. After the text "This is a new file." add "I have made a change".
3. Save the file.
   In PHP Explorer view, the file and project will have a ">" suffix, indicating that a change has been made which has not yet been committed.
4. So far, the changes have only been saved in your workspace. In order to compare the local file to which you have made changes with the one sitting in the CVS repository, right-click the file in the PHP Explorer view and select **Team | Synchronize with Repository**.
5. A Text Compare dialog will open showing the local file you have just made changes to (in the left-hand window) as compared to the file in the repository (in the right-hand window).
   The change you have made will be highlighted.



Committing Changes

Once you have edited your file and compared it to the one in the repository to ensure that the changes are correct, you can commit your changes.

To commit your changes to the repository:

1. In PHP Explorer View, right-click your file and select **Team | Commit**.
2. A Commit dialog will open.
   Enter the comment "I have made changes to CVSFile1." and click **Finish**.
3. Your changes will now be committed to CVS and all users will be able to access the newly updated file.

Note:

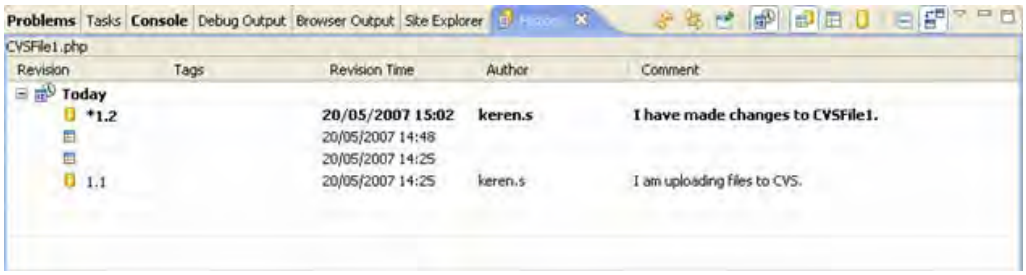
If you had made changes to a number of files, you can use the Synchronize view, within the Team Synchronizing Perspective, to view and commit all your changes at once.

Replacing Files with Older Versions

Using CVS's version control system, users can revert back to older versions of files if incorrect or irrelevant changes have since been made.

This procedure demonstrates how to replace your file with an older version from CVS:

1. In PHP Explorer view, right-click your CVSFile1 and select **Replace with | History**.
2. A History view will be displayed, listing the times that the file has been committed.
   The changes made on CVS (as opposed to local workspace changes) will be shown with a repository icon.
   As committed changes have been made twice (once during the original upload and once when you edited the file), 2 CVS revisions should be listed with their relevant comments.



3. To open the Text Compare view to see the previous version of the file, double-click the second CVS revision in the list (containing the comment "I am uploading files to CVS.") This is the original state the file was in when it was first uploaded.
4. To return the file to its previous state, select the second revision again, right-click and select **Get Contents**.
5. Your file will be replaced with the contents of the previous version of the file (without the line "I have made a change.")

Deleting Files from CVS

You can delete a file from the CVS repository by first deleting it from your workspace and then committing the deletion.

This procedure demonstrates how to delete a file from CVS:

1. In PHP Explorer view, right-click CVSFile2 and select **Delete**.
2. Click **Yes** when asked to confirm the deletion.
3. In PHP Explorer view, right-click MyCVS Project and select **Team | Synchronize**.
4. The Team Synchronizing Perspective will open.
   The file you have deleted will be displayed with a purple arrow with a minus sign.

5. Right-click the file and select **Commit**.
6. A Commit Deletion dialog will open.
   Enter a comment "Deleting CVSFile 2."
7. Click **OK**.
8. The file will be deleted from CVS.

Note:

This action will delete the file from the CVS repository and not just from your workspace. This file will no longer be accessible by any users.

Working with SVN

The purpose of this tutorial is to teach you how to work with the SVN source control system. You will learn how to configure your SVN repository, upload projects and files to it, check out (import) projects and files from it and commit changes which you have made to files.

Contents:

Purpose
Adding an SVN Repository
Sharing Projects
Checking Out Projects from SVN
Adding Files to Existing Projects
Making Changes, Comparing Changes and Committing Changes
Replacing Files with Older Versions
Deleting Files from SVN

Purpose

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

Adding an SVN Repository

Before you can add projects to or export projects from SVN, you must define your SVN repository settings.

Note:
To access a repository, make sure that an SVN server is already configured.

To add a new SVN repository:

1. Open the SVN perspective by going to **Window | Open Perspective | Other | SVN Repository Exploring**.
2. In the SVN Repositories view, click the Add SVN Repository button on the view's toolbar -or- right-click within the SVN view and select **New | Repository Location**. The Add SVN Repository dialog will open.
3. Enter the information required to identify and connect to the repository location:
   - URL - The URL on which your repository is located.
   - Authentication - The user name and password you use to connect to SVN.
4. Click **Finish**.
5. Your SVN repository will now be added to the SVN Repository view.

Sharing Projects

Through SVN, projects can be shared and worked on by numerous team members.

The following steps demonstrate how to upload a project to your SVN repository location:

1. Create a new PHP project called "MySVN Project".
2. Within the project, create a PHP file called "SVNFile1" with the following code:

```
<?php
//This is a new file
?>
```

3. In PHP Explorer View, right-click your project and select **Team | Share Project**.
   A Share Project dialog will open.
4. From the repository list, select SVN and click **Next**.
5. Select 'Use existing repository location', and select your repository from the list.
6. Click **Finish**.
7. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click **Next**. (Mark the Save Password checkbox to ensure that this screen does not reappear.)
8. A Commit dialog will open. Enter the comment "I am uploading files to SVN." and click **OK**.
9. In PHP Explorer View, your project will now have a repository icon , indicating that it is in SVN.
10. Once you have committed your files, other team members will be able to access and change the files.

The instructions below explain how users can check out (import) projects from SVN, edit them and upload their changes.

Checking Out Projects from SVN

Once projects are placed on the SVN repository, they can be checked out (imported) by anyone with access to that repository.

The following steps demonstrate how to check out (import) projects from SVN into your workspace:

1. Delete the 'MySVN' project from from your workspace in order to simulate being a new user who has not previously had access to this file.
   Note: Deleting the project from your workspace will not delete it from SVN.
2. Go to File Menu and select **Import | Projects from SVN**.
3. Click **Next**.
4. Select your repository.
5. Click **Next**.
6. A 'Select Resource' dialog will appear. Expand the nodes until you see your project (by default, this will have been placed under 'Trunk'.
7. Select your project and click **Finish**.
8. A 'Check Out As' dialog will appear.
   Leave the 'Check out as a project with the name specified' option marked and click **Finish.**

The project will now be imported into your workspace.

Note that the project will have an SVN repository icon in your PHP explorer view.
Now that you have imported a project from SVN into your workspace, you can add files, edit existing files and commit your changes to the SVN repository. (See below).

Adding Files to Existing Projects

You can add files to existing projects in the SVN repository and commit them so that other users can access them.

The following steps demonstrate how to add and commit a file into an existing project:

1. In your SVNProject, create a new PHP file, called "SVNFile2" with the following code:

```
<?php
//Another new file
?>
```

2. Save the file.
3. In PHP Explorer View, select the file, right-click and select **Team | Commit**.
4. A Commit dialog will open.
   Enter a comment "Another new file added." and click **OK**.

The file will be committed to SVN and will be accessible by other users.

Making Changes, Comparing Changes, and Committing Changes

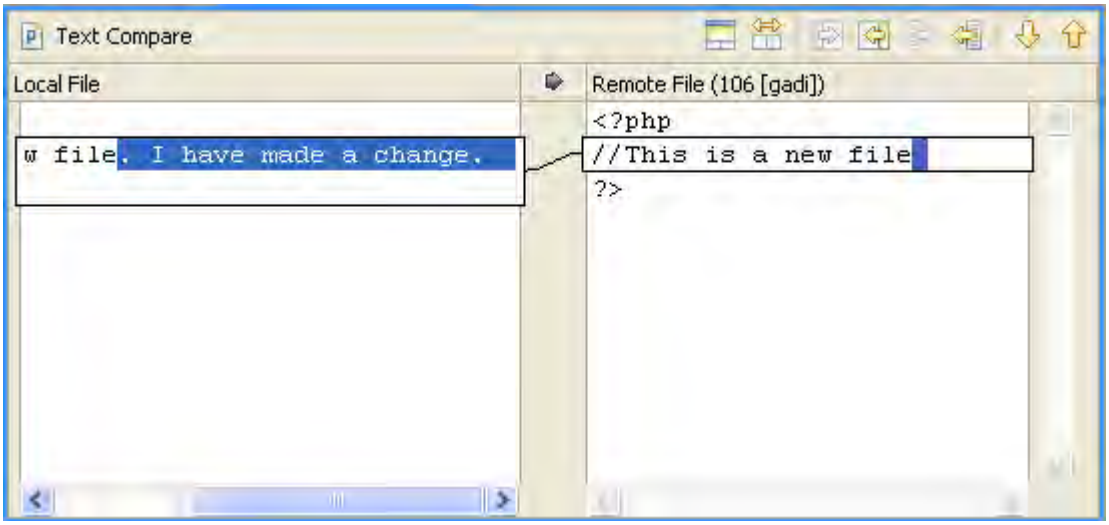
Once files are stored on SVN, you and all other team members can make changes to the files and commit them. Before committing changes you have made to a file, you can compare the file stored locally in your workspace to the file stored on the SVN repository.

Making and Comparing Changes

The following procedure demonstrates how to make changes to files and comparing local files to files in the repository:

1. Open SVNFile1 in your SVNProject.
2. After the text "This is a new file.", add "I have made a change".
3. Save the file.
   In PHP Explorer view, the file and project will have a ">" suffix, indicating that a change has been made which has not yet been committed.
4. So far, the changes have only been saved in your workspace. In order to compare the local file to which you have made changes with the one sitting in the SVN repository, right-click the file in the PHP Explorer view and select **Team | Synchronize with Repository**.
5. Click **Yes** when asked whether to open the Team Synchronizing perspective. (Check the 'Remember my decision' box to prevent this prompt from appearing in the future.)

A Text Compare dialog will open showing the local file you have just made changes to (in the left-hand window) as compared to the file in the repository (in the right-hand window).
The change you have made will be highlighted.



In addition, the Synchronize view will have opened in the left hand-side, displaying the file to which you have made changes. The file will have a grey arrow icon indicating that changes have been made which need to be synchronized with the repository.

Committing Changes

Once you have edited your file and compared it to the one in the repository to ensure that the changes are correct, you can commit your changes.

To commit your changes to the repository:

1. From the Synchronize view, click the 'Commit All Outgoing Changes' button.
   -Or- In PHP Explorer View, right-click your file and select **Team | Commit**.
2. A Commit dialog will open.
   Enter the comment "I have made changes to SVNFile." and click **Finish**.

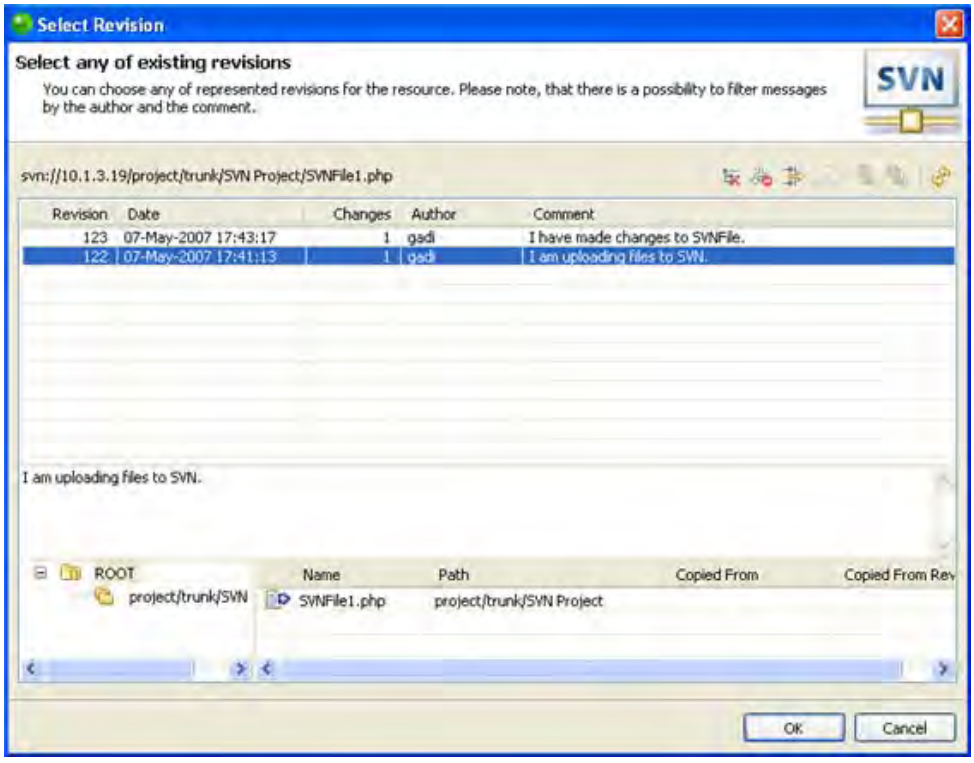
Your changes will now be committed to SVN and all users will be able to access the file.

Replacing Files with Older Versions

Using SVN's version control system, you can revert back to older versions of files if incorrect changes have since been made.

This procedure demonstrates how to replace your file with an older version:

1. In the PHP Explorer view, select your SVNFile1 and from the right-click menu select **Replace with | Revision**.
2. A Replace with Revision dialog will open.
   Select Revision and click **Select**.
3. A “Select Revision” dialog will open, listing the various times the file has been committed (revisions).
   As committed changes have been made twice (once during the original upload and once when you edited the file), 2 revisions should be listed with their relevant comments.



4. Scroll between the two revisions.
   To return the file to its previous state, select the second revision in the list (containing the comment "I am uploading files to SVN.") This is the original state the file was in when it was first uploaded.
5. Click **OK**.

6. You will be returned to the Replace with Revision dialog, which will now have a revision number in the Revision box.
7. Click **OK**.
8. You will be prompted to confirm that you would like to replace the revision. Click **Yes**.

Your project will be reverted to the old one and the line "I have made a change" will be removed from SVNFile1.

Deleting Files from SVN

You can delete a file from the SVN repository so that the file will no longer be available to any users.

This procedure demonstrates how to delete a file from SVN:

1. Open the SVN Repositories view.
2. Expand the nodes to find your project.
3. Right-click the file you would like to delete and select **Delete**.
4. A Commit Deletion dialog will open. Enter a comment if required.
5. Click **OK**.

The file/project will be deleted from your SVN repository.

Note:

This action will delete the file from the SVN repository and not just from your workspace. This file will no longer be accessible by any users.

See the Subversive User Guide for more information on the SVN plugin.

Working with the Debugger

The purpose of this tutorial is to teach you how to debug files and applications both remotely and locally in order to gain maximum efficiency and accuracy from your files and projects.

Contents:

Purpose and Usage
Debugging Local PHP Files (PHP Scripts)
Debugging PHP Applications (PHP Web Pages)

Purpose and Usage

Zend Studio's Debugging feature can detect and diagnose errors in PHP code situated locally or on remote servers. The debugger allows you to control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables.

Debugging should be used at stages where your scripts and applications are formed sufficiently to be tried and tested.

Zend Studio includes several types of debugging:

- Locally Debugging PHP Scripts - Debugging PHP files using Zend Studio's internal PHP Executable debugger.
- Remotely Debugging PHP Scripts - Debugging PHP files using your server's debugger.
- Debugging PHP Web Pages - Debugging files, applications and projects on your server, using the local or server copies of your files.
- Debugging URLs - Debug applications on your server by entering a URL.
- Toolbar Debugging - Debug files and applications directly from your browser using the Zend Studio Toolbar.

Debugging Local PHP Files (PHP Scripts)

PHP files (PHP Scripts) on your workspace can be debugged using either Zend Studio's internal debugger or your server's debugger. Using your server's debugger, you can test the execution of the file on your server's environment. This is especially relevant if your server's loaded extensions are different to Zend Studio's internal server.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

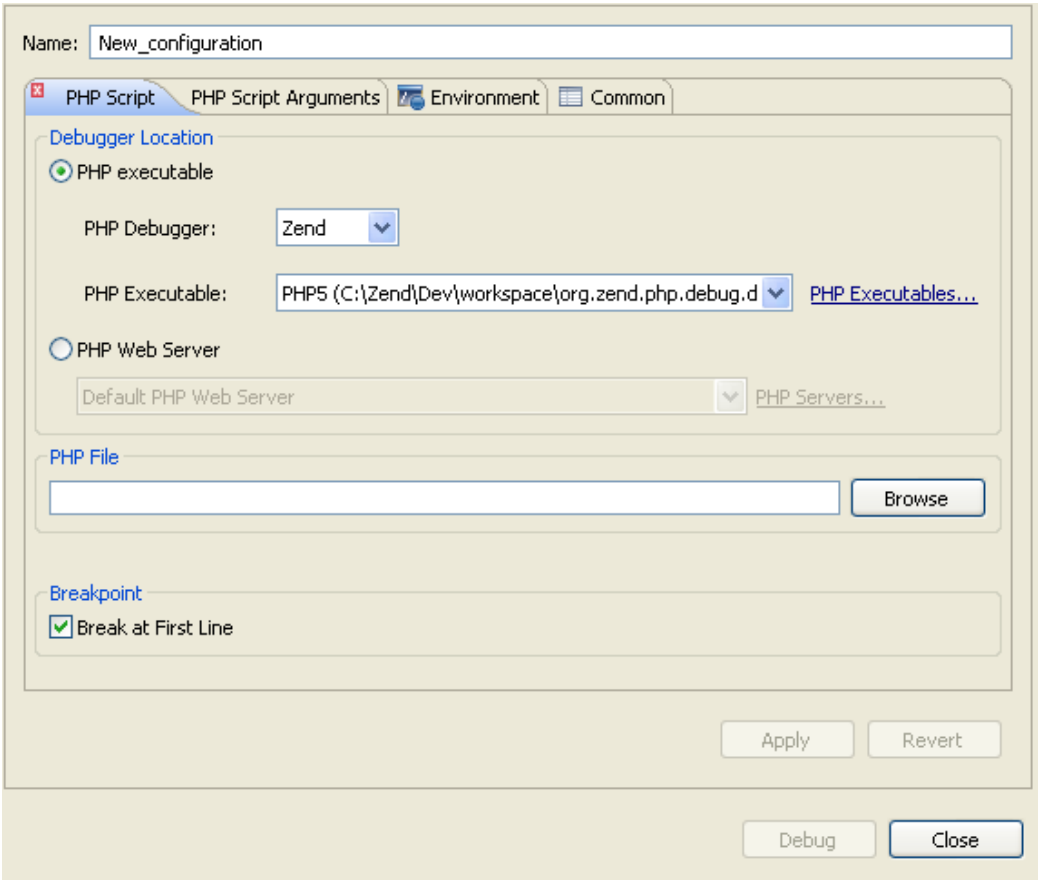
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

Debugging a PHP File Locally

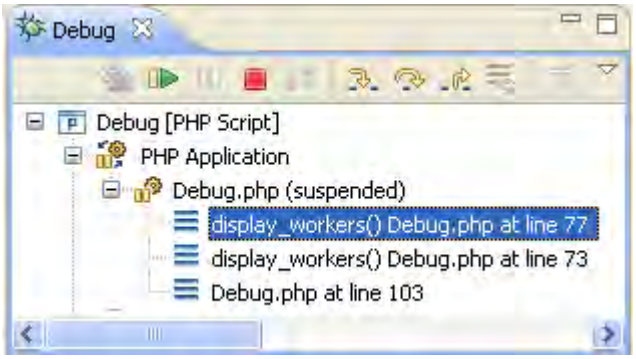
Files can be debugged using Zend Studio's internal debugger.

This procedure demonstrates how to debug a file using the internal debugger or your server's debugger:

1. Inside a PHP project, create a PHP file, called "debug", and copy-paste the example code into it.
   (See the "Working with the Debugger" Tutorial in Zend Studio's Online Help for the example code.)
2. Set a breakpoint at line 103 by double-clicking the marker bar to the left of the editor window.
   A blue ball will appear.
3. Save the file.
4. Click the arrow next to the debug button on the toolbar and select **Debug Configurations** -or- select **Run | Debug Configurations...** from the main menu -or- right-click the file in PHP Explorer view and select **Debug Configurations....**
   A Debug dialog will appear.
5. Double-click the PHP Script option to create a new debug configuration.

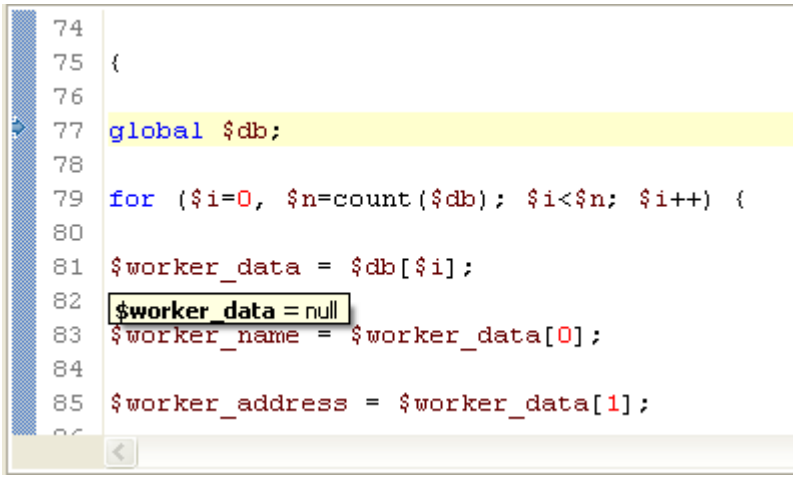


6. Enter a name for the new configuration.
7. To debug the file using Zend Studio's PHP Executable debugger, select the PHP Executable option in the Debugger Location category.
   -Or- To debug the file using your server's Debugger, select the PHP Web Server option under the Debugger Location category and select your server from the drop-down list.
   If you have not configured a server, click the PHP Servers link to be taken to the PHP Servers Preferences page.
8. Under PHP File, click **Browse** and select the "debug" file.
9. Ensure that the 'Break at First Line' Breakpoint checkbox is selected.
9. Click **Apply** and then **Debug**.
10. Click **Yes** if asked whether to open the PHP Debug Perspective.
11. A number of views will open with information about your script.
12. The Debug View is where the debug stack trace is displayed and the debugging process can be monitored and controlled.



The debugging process will currently have stopped where your first <?php label appears.

13. Click the Resume icon to continue to the breakpoint.
14. Click Step Into. The Debugger goes into the function defined in line 103 and advances to line 77.
15. The Variable view will now display various information about the relevant variables and parameters through which the function was reached.
16. In the editor window, place and hold the cursor over $worker_name, $worker_address, and $worker_phone. A tooltip appears displaying the variable values.



17. Click Step Return. The cursor returns to line 103.
    The Debug Output view will display the HTML output created up until the breakpoint, while the Browser Output view will show the current output to a browser.
18. In the Debug view, click Resume until the debugging process is terminated.

Notice that as the debugging process progresses, the Debug Output and Browser Output displays are updated.

19. The console view will display any errors or warnings about your script. In this case, it will display a Notice about an undefined variable on line 105.
20. Click on the PHP Perspective icon to return to normal editing mode.
21. To run the debugging process again, click the arrow next to the debug icon on the toolbar and select your configuration -or- select **Debug Configurations...** and double-click your configuration from the "Debug" dialog.
    Clicking the debug icon will debug the last executed launch configuration.

Debugging a PHP File Remotely

Files can also be debugged remotely using the Debugger situated on your server. Use this function if you want to test the execution of the file on your production environment. This is especially relevant if your server's loaded extensions are different than the local environment..

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

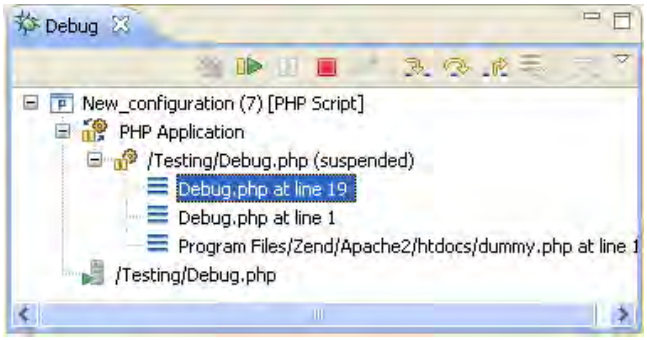
This procedure demonstrates how to debug a file remotely:

(If you have already created the file following steps 1 - 3 under "Debugging a PHP File Locally", above, skip to step 4.)

1. Inside a PHP project, create a PHP file, called "debug", and copy-paste the example code into it.
2. Set a breakpoint at line 103 by double-clicking the marker bar to the left of the editor window.
   A blue ball will appear.
3. Save the file.
4. Click the arrow next to the debug button on the toolbar and select **Debug Configurations...** -or- right-click the file in PHP explorer or within the file's editor window and select **Debug As | Open Debug Dialog**.
   A Debug dialog will appear.
5. Double-click the PHP Script option to create a new debug configuration.
6. Enter a name for the new configuration.
7. In your new configuration window, select PHP Web Server under the Debugger Location category.
8. Click PHP Servers to add your server.
   If you already have a server configured, skip to step 12.
9. In the PHP Servers dialog, click **New**.
10. Enter the server's name and its document root URL.
11. Click **Finish** and **OK** to add your server and return to the "debugging" dialog.
12. Select your server from the drop-down list.
13. Under PHP File, click **Browse** and select the "debug" file.

14. Mark both checkboxes under the Breakpoint category.
15. Click **Apply** and **Debug**.

    The Debug Perspective will open.
16. In the Debug view, click Resume until the debugging process is terminated.
17. **The output in the views will provide the following information:**
    - Debug View – Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out off) certain functions.



    - Variables – Will display the various variables in your script.
    - Breakpoints – Will display the breakpoints you have entered
    - Parameter Stack – Will display the parameters through which functions are reached.
    - Editor Window – will display the code at the relevant sections, according to which line is selected in the Debug View window.
    - Debug Output – Will show the textual output of the script. This will be updated as the debugging process continues.
    - Browser output - Will show the output of the script to a browser. This will be updated as the debugging process continues.

- Console View – Will display any error and warning messages.
- Tasks – If you had added any tasks to your script, these would be displayed here.

18. Click on the PHP Perspective icon to return to normal editing mode.
19. To run the debugging process again, click the arrow next to the debug icon on the toolbar and select your configuration -or- select Debug and double-click your configuration from the Debug dialog.

Debugging PHP Applications (PHP Web Pages)

Zend Studio also allows you to debug applications, projects or files that are already on the server. You can debug either the local (Workspace) copy of files or the server copy of files.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

This procedure demonstrates how to debug applications on a server:

1. Inside a PHP project, create a new PHP file, called "form1", with the following code:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

2. Create a second PHP file, called "welcome", with the following code:

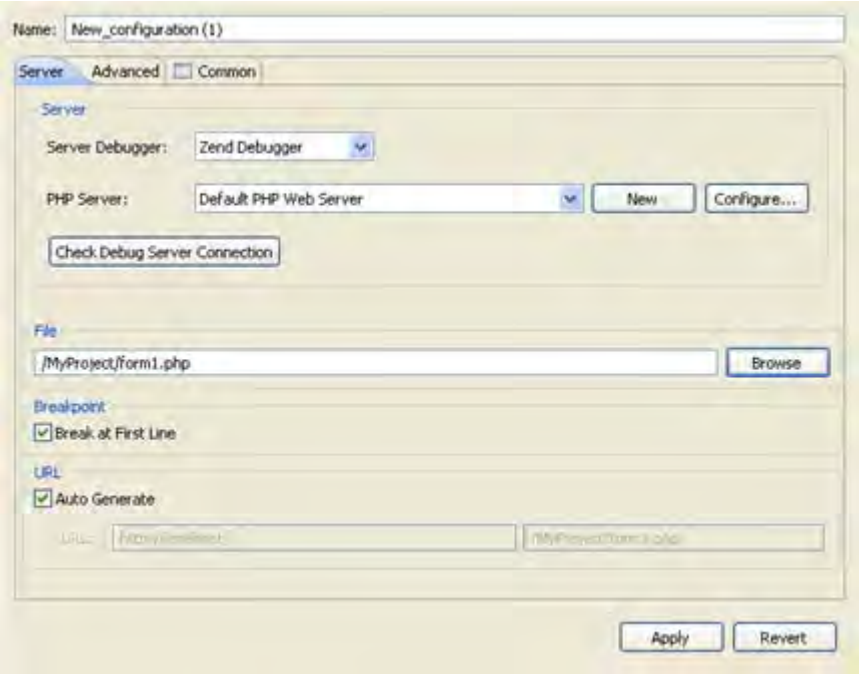
```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

3. Save both files and copy them to your server.
4. Click the arrow next to the debug button on the toolbar and select Debug Configurations... -or- right-click the file in PHP explorer or within the file's editor window and select **Debug as | Debug Configurations....**
   A Debug dialog will appear.
5. Double-click on the PHP Web Page option to create a new debug configuration.
6. Enter a name for the new configuration.

7. Select the Zend Debugger from the Server Debugger drop-down list.
8. Select your server from the drop-down list.
   If you have not configured a server, click the PHP Servers link to be taken to the PHP Servers Preferences page.
9. Under the File/Project category, click **Browse** and select the "form1" file. This will be the file from which the debugger will start debugging (the 'debug target'.)
10. Ensure that the URL pointing to the file location is correct.
    If this is not correct, unmark the Auto Generate checkbox and manually change the URL.

Note:

You can choose whether the file content will be taken from the local copies of the files or from the files located on your server. To select the file's Source Location, select the 'Advanced' tab and select the relevant option under the 'Source Location' category.

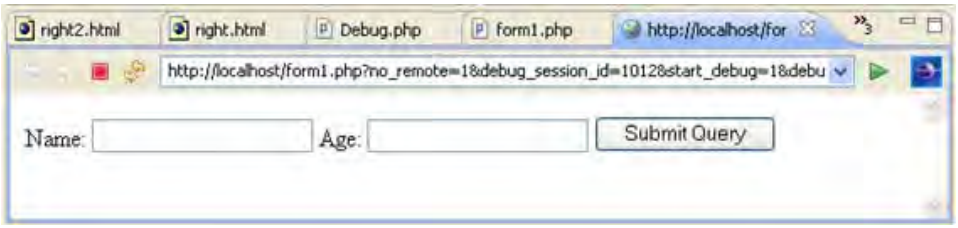


11. Click **Apply** and then **Debug**.
12. Click **Yes** when asked whether to open the PHP Debug Perspective.
13. The Debug Perspective will open with several views relevant to the debugging process (See 'PHP Debug Perspective' for more information on the different views.)
14. In the editor view, you will see the code for the "form1" file.
15. In the Debug view, click **Resume** to resume the debugging process.
16. The browser output will display a form asking you to enter your Name and Age.
17. Select the browser view (tabbed with the editor window). This will display the output

of your script in a browser in 'real time'.

Note that this is different from the Browser Output window.

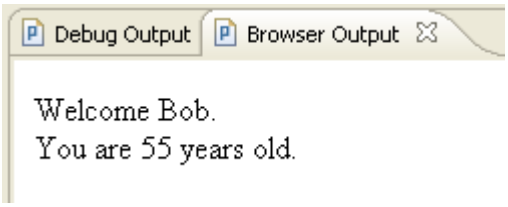
18. In the browser view, enter your Name and Age and click **Submit Query**.



19. Another editor tab will open, with the script from the welcome.php file.
20. In the Debug view, click **Resume** to resume the debugging process.
21. The browser output and browser views will display the final result of your application:

    "Welcome [Name].

    You are [Age] years old."



22. The debugging process will have terminated.
23. Click on the PHP Perspective icon to return to normal editing mode.
24. To run the debugging process again, click the arrow next to the debug icon on the toolbar and select your debugging configuration.

Working with Refactoring

The purpose of this tutorial is to teach you how to refactor your code to easily rename and move files and elements without damaging your projects or severing the link between referenced items, as well as creating the necessary links between PHP elements in separate files.

Contents

Purpose and Usage

Renaming / Moving Files

Renaming Elements within a File

Purpose and Usage

The Refactoring feature allows you to:

- Rename and move files and elements within those files, while maintaining the links between the items. Once an element or file has been renamed or moved, all instances of that item within the project will be automatically updated to reflect its new name / location.
- Organize Includes so that elements from one file can be referenced in another file.

Refactoring should be used when you are reorganizing your project and changing names and locations of files and elements.

Note:

Refactoring will only work from within PHP Explorer view and not from Navigator view.

Renaming / Moving Files

Renaming a file

Renaming a file will result in the automatic renaming of all instances where that file is referenced within the project.

This procedure demonstrates how to Rename a file:

1. Create a PHP file, called RenFile1, with the following code:

```
<?
$a = 5;
?>
```

2. In the same project, create a second PHP file, called RenFile2, with the following code:

```
<?
require("RenFile1.php");
$a = 8;
?>
```

Note:

Ensure you delete the PHP tags that are inserted by default when creating a new PHP file before pasting in the code.

3. Save both files.
4. In PHP Explorer view, right-click RenFile1 and select **Refactor | Rename &endash**; or- select it and go to **Refactor | Rename** from the Main Menu.
   A Rename File dialog will appear.
5. In the Rename File dialog box, rename RenFile1 to RenFile3.
6. Check the "Update references" box and click **Preview**.
7. In the Preview window, scroll through the changes and note that, as well as the name of the file itself being changed, the reference to the file in RenFile2 will also have been changed.
8. Press **OK** to apply changes.

The reference to RenFile1 in RenFile2 will have been updated to the following in order to reflect the changes in the file name:

```
require("RenFile3.php");
```

Moving a file

Moving a file will result in the automatic updating of all instances where that file is referenced within the project to reflect its change of location.

This procedure demonstrates how to move a file:

1. Create RenFile1 and RenFile2 as described in steps 1 to 3 under "Renaming a File", above.
2. Within the same project, create an additional folder called RenFolder.
3. In PHP Explorer view, right-click RenFile1 and select **Refactor | Move** -or- select it and go to **Refactor | Move** from the Main Menu.
4. In the Move dialog, select RenFolder.
5. Click **Preview**.

   The Preview windows will display the changes that the move will apply to your script. Note that RenFile1's new location will automatically be updated in the reference to it in RenFile2.
6. Click **OK** to execute the Move.

Renaming Elements within a File

All PHP Elements (e.g. classes, interfaces, functions, methods) can also be renamed and refactored from within the editor window.

This procedure demonstrates how to use the Rename Elements feature:

1. Create 2 files (RenFile1 and RenFile2) as described in steps 1 to 3 under "Renaming a File", above.
2. In the editor window of RenFile2, highlight the variable "a" on line 3.
3. Right-click and select **Refactor | Rename** -or- click **Alt+Shift+R.**
4. In the Rename Global Variable dialog box, rename the variable to b.
5. Check the "Update textual occurrences" box and click **Preview**.
6. In the Preview window, scroll through the changes and note that occurrences of variable "$a" will be changed in RenFile1 as well as in RenFile2.
7. Press **OK** to apply changes.

Working with the Profiler

The purpose of this tutorial is to teach you how to profile files and applications in order to gain maximal efficiency in the execution of your script.

Contents

Purpose and Usage

Profiling PHP Scripts

Profiling PHP Web Pages

Purpose and Usage

Zend Profiler detects bottlenecks in scripts by locating problematic sections of code. These are scripts that consume excessive loading-time. The Profiler provides you with detailed reports that are essential to optimizing the overall performance of your application.

Zend Studio includes five types of profiling:

- Locally Profiling PHP Scripts - Profiling PHP files using Zend Studio's internal PHP Executable debugger.
- Remotely Profiling PHP Scripts - Profiling PHP files using your server's debugger.
- Profiling PHP Web Pages - Profiling files, applications and projects on your server, using the local or server copies of your files.
- Profiling URLs - Debug applications on your server by entering a URL.
- Toolbar Profiling - Debug files and applications directly from your browser.

Profiling PHP Scripts

Files can be profiled using either Zend Studio's internal debugger or your external (remote) server.

Use the remote profiling function if you want to test the execution of the file on your server's environment. This is especially relevant if your server's loaded extensions are different to Zend Studio's internal server.

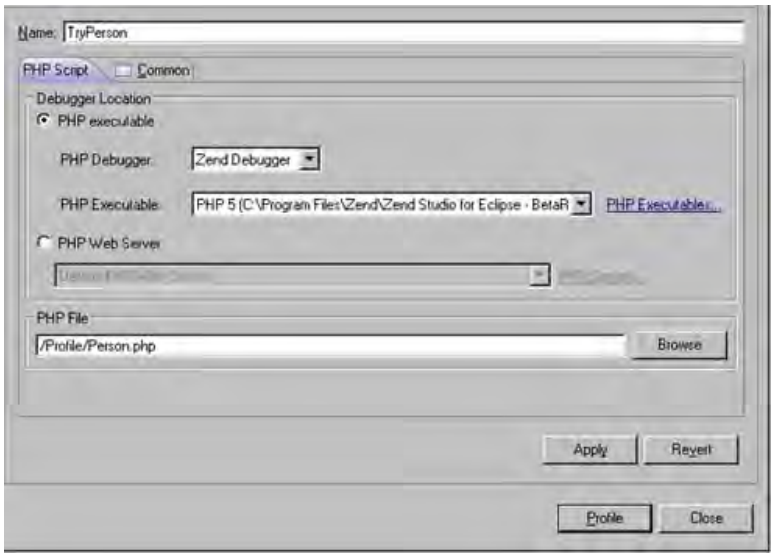
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

The following procedure demonstrates how to profile a PHP Script, either locally or remotely:

1. Create a file, called Person, and copy-paste the example code into it.
   (See the "Working with the Profiler" Tutorial in Zend Studios Online Help for the example code.)
2. Create a second file, called tryPerson, and copy-paste the example code into it.
   See the "Working with the Profiler" Tutorial in Zend Studio's Online Help for the example code.
3. Save both files.
4. Click the arrow next to the Profile button on the toolbar and select **Profile Configurations...**. -or- from the main menu go to **Run | Profile Configurations...** -or- right-click in PHP Explorer view and select **Profile Configurations....**
5. A Profile dialog will appear.

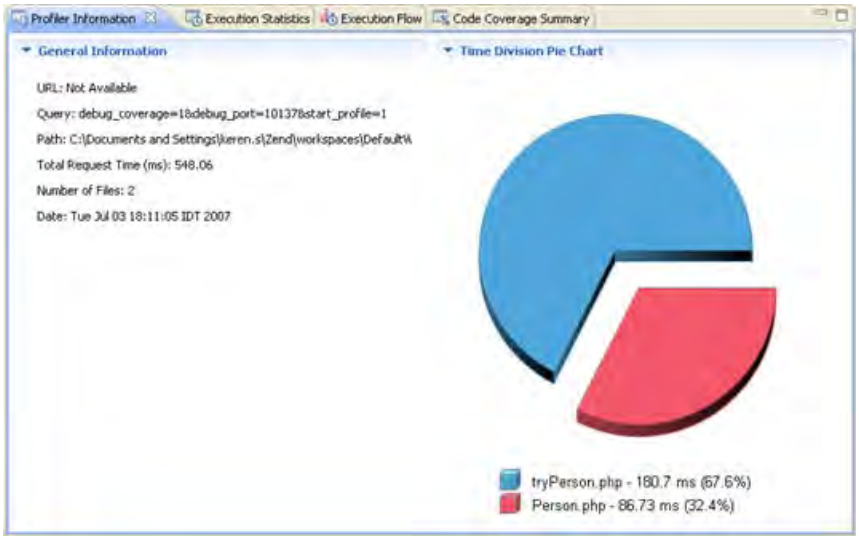


6. Double-click the PHP Script option to create a new Profile configuration.
7. Enter a name for the new configuration.
8. To profile the file locally using Zend Studio's internal debugger, select the PHP Executable setting under the Debugger Location category and select the required PHP executable (PHP 4 or 5).
   To profile the file remotely on your server using the Zend Debugger installed on your server, select the PHP Web Server option and select your server from the drop-down list. (If you have not yet configured a server, click the PHP Servers link and follow the instructions under Adding PHP Servers.)

9. Under PHP File, click **Browse** and select the "TryPerson" file.
10. Click **Apply** and then **Profile**.
11. A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.
    Click **Yes**. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with the following information:

- **Profiler Information** - provides general information on the profiling duration and date, number of files constructing the requested URL and more. In addition, it displays a Time Division Pie Chart for the files in the URL.
  The right side displays time division in a pie chart and the left side provides the following information:
- URL - The URL analyzed
- Query - The specific query parameters
- Path - The exact location of the first file called
- Total Request Time - Total process time of the entire page
- Number of Files - Number of files composing the page
- Date - Date and time that the profiling took place



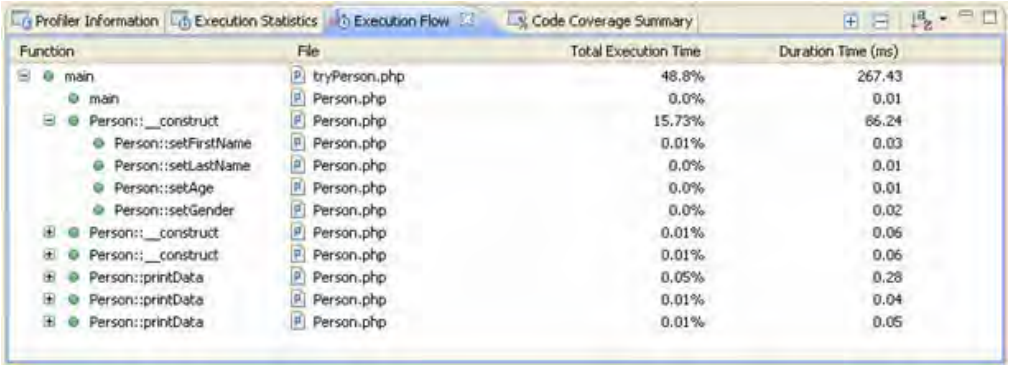
- **Execution Statistics** - Displays the list of files constructing the URL and detailed information on functions in the files. The window contains statistics relevant to each function:
- Function - The name and location of the function.
- Calls Count - The number of times that the function was called.

- Average Own Time - The average duration without internal calls.
- Own Time(s) - The net process duration without internal calls.
- Others Time(s) - Time spent on calling other files.
- Total Time(s) - The total process duration including internal calls.

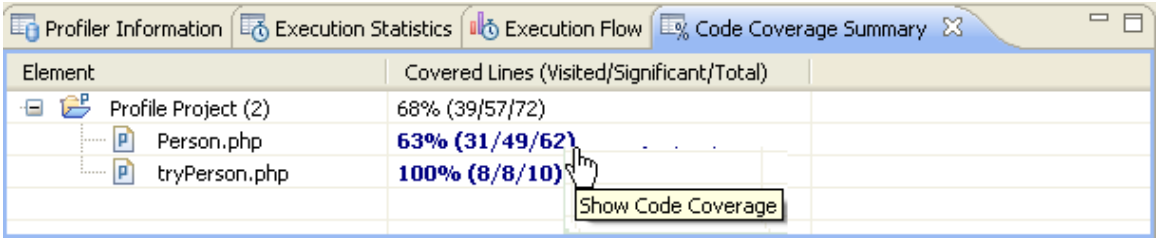
Note:

Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.

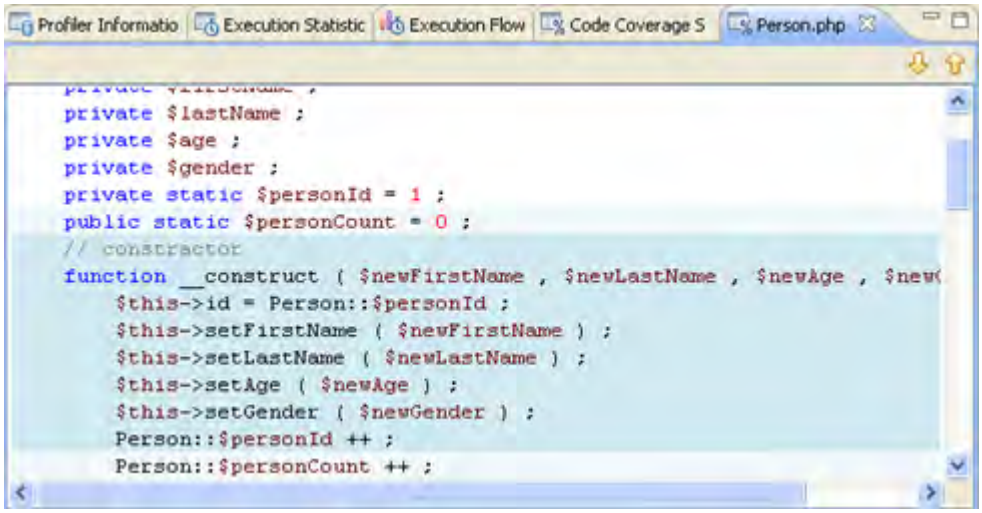
- **Execution Flow** - Shows the flow of the execution process and summarizes percentages and times spent on each function.
- Function - Function name
- File - The file in which the function is located
- Total Execution Time - Percent of time taken per function.
- Duration Time - Time taken per function. In milliseconds.



- **Code Coverage Summary** - Summary of how many lines of code were covered.
- Element - The file / folder that was called.
- Covered Lines (Visited / Significant / Total) - Percentage of lines covered within each file. (Visited = Number of lines covered / Significant = number of lines excluding comments and declarations/ Total = Total number of lines in the file.)



Clicking on the 'Covered lines' percentages will open an editor containing the debug file, with the covered lines highlighted:



Profiling PHP Web Pages

Zend Studio also allows you to profile whole applications, projects or collections of files that are already on the server.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

The following steps demonstrate how to profile a file on an external server:

1. Create a file, called Person, and copy-paste the example code into it.
   See the "Working with the Profiler" Tutorial in Zend Studio's Online Help for the example code.
2. Create a second file, called tryPerson, and copy-paste the example code into it.
   See the "Working with the Profiler" Tutorial in Zend Studio's Online Help for the example code.
3. Save both files.
4. Copy them to your server.
5. Click the arrow next to the Profile button on the toolbar and select **Profile Configurations...** –or- right-click the file in PHP explorer or within the file's editor window and select **Profile As | Open Debug dialog**.
   A Profile dialog will appear.
6. Double-click on the PHP Web Page option to create a new profile configuration.
7. Enter a name for the new configuration.
8. Select the Zend Debugger to from the Server Debugger drop-down list.
9. Ensure that your server is selected from the list.
   If you have not configured a server, click **New** and enter:
   i. Your server's name.
   ii. The URL of its document root.
10. Under the File/Project category, click **Browse** and select the "tryPerson" file. This will be the file from which the profiling process will start.
11. Click **Apply** and then **Profile**.
    A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.

12. Click **Yes**. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with profiling information.

See PHP Perspectives and Views for details on the information displayed in the profiling perspective.

Working with PHPUnit Testing

The purpose of this tutorial is to teach you how to create and run PHPUnit tests on your code. You will learn how to create and run single unit test cases and test suites containing a number of test cases.

Contents:

Purpose and Usage
Creating a PHPUnit Test Case
Running your PHPUnit Test Case
Analyzing Errors
Creating and Running a PHPUnit Test Suite
Generating a PHPUnit Test Report

Purpose and Usage

Unit testing is a procedure to test your code to ensure that individual units of source code are working properly and that the right output is being generated. Tests can be run on all or some functions within files, meaning that tests can be conducted before the file has been fully developed. Each test case should be independent of others to ensure that test results can pinpoint the location of the error.
Running unit tests can ensure that your code is stable and functioning correctly, and can help you to diagnose errors.

Creating a PHPUnit Test Case

Zend Studio will automatically create test case files which can be run in order to check the functionality of your code.

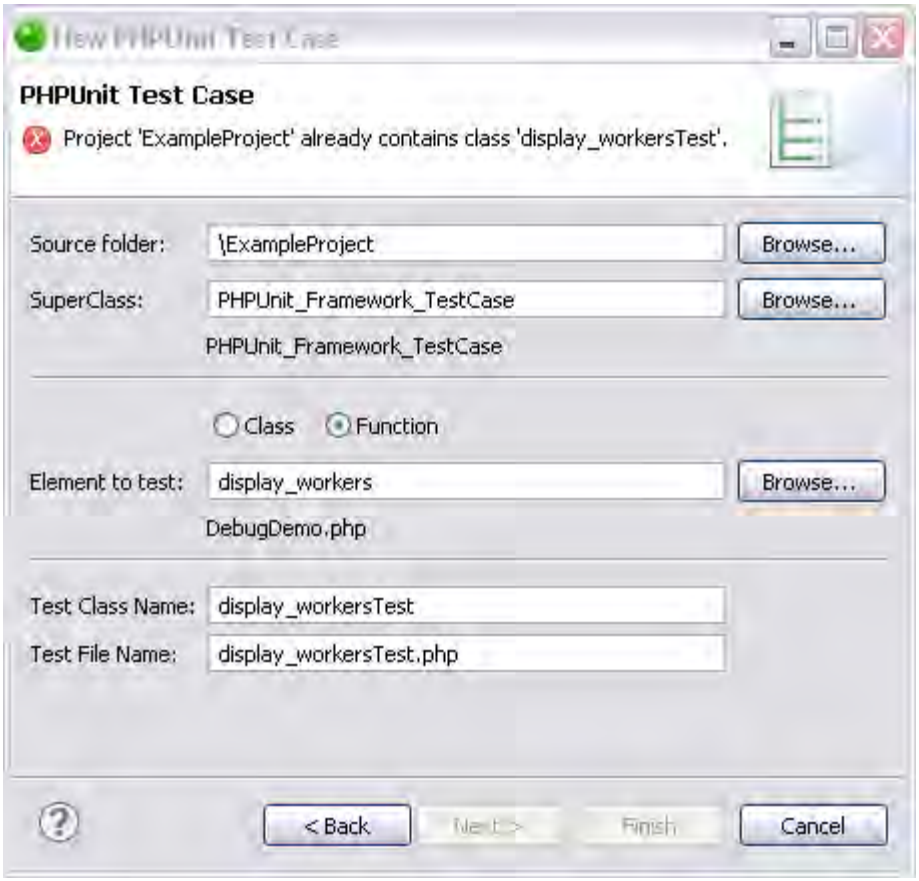
The following steps demonstrate how to create a PHPUnit Test Case:

1. Create a new PHP file, called "Calculator", and copy-paste the following code into it:

```
<?php
class Calculator {
public function add($a, $b) {
return $a + $b;
}

public function multiply($a, $b) {
return $a * $b;
}

public function divide($a, $b) {
if($b == null) {
throw new Exception("Division by zero");
}
return $a / $b;
}

public function subtract($a, $b) {
return $a - $b;
}
}
?>
```

2. Save the file.
3. In PHP Explorer view, right-click the calculator file and select **New | Other | PHP | PHPUnit | PHPUnit Test Case**.

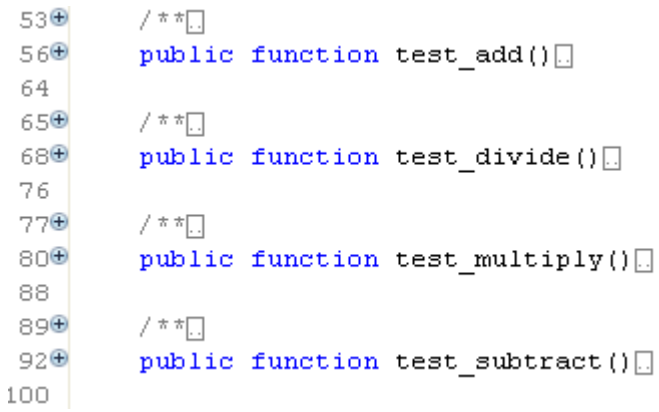
   The PHPUnit Test Case dialog will open.



4. The relevant information will have already been entered (if not click the link in the dialog to add the PHPUnit to the include path and this will populate the "Element to Test" list). Note that a new file will be created called CalculatorTest.php
   A warning will also appear stating that the PHPUnit is not the include class of your project.
5. To add it to the include path, click the underlined **Click here** link at the bottom of the dialog screen.
   Once it has been clicked, the link and the warning message will disappear.
6. Click **Finish** to create your test case.

A CalculatorTest file will have been added to your project in PHP Explorer. This will contain tests for your original "calculator" file.

Note that all functions (add, multiply, divide and subtract) in the original "Calculator" file will have a corresponding test function in the "CalculatorTest" file:



Note:

Test functions will have been created, but parameters need to be entered in order for the test to run effectively. For more on this, see "Running your Unit Test Case", below.

Running your PHPUnit Test Case

Having created your PHPUnit Test Case, you will now need to customize it by entering relevant parameters to be checked before being able to run your test.

To configure and run your test case:

1. In the CalculatorTest file, expand public function test_add node.
2. Note that a function has been created but no parameters have been inserted. You will have to manually enter the relevant parameters to be tested and the predicted results.
3. Delete the following code:

```
// TODO Auto-generated CalculatorTest->test_add()
$this->markTestIncomplete("add test not implemented");
$this->Calculator->add(/* parameters */);
```

4. This is the default test which will return a "test not implemented" result if the test case is run.
5. Replace the above code with the following:

```
$this->assertEquals($this->Calculator->add(1, 2), 3);
```

6. The numbers 1 and 2 indicate that when the test case is run, the parameters 1 and 2 will be entered into the 'add' function in your Calculator file (i.e. the test will try to add 1 + 2). The last number (3) indicates that the expected result is 3. If the result is something other than 3, the test will report a failure for this function.
7. Save the file.
8. To run the unit test, click the arrow next to the Run button on the toolbar and select **Run As | PHPUnit Test** –or- go to Run Menu and select **Run As | PHPUnit Test** .

   Or- to debug the PHPUnit Test Case, click the arrow next to the debug button on the toolbar and select **Debug As | PHPUnit Test** –or- from the Main Menu, go to Run and select **Debug As | PHPUnit Test** .

   The unit test will be run and a PHP Unit view will open.

   As the test is run, the parameters you have configured will be entered into the relevant functions in the Calculator file to test whether the correct result is outputted according to the expected results you specified.
9. Four tests will be displayed - one for each calculator function - which should have passed successfully, as indicated by the green tick icon .

   Note that the other three functions (divide, multiply and subtract), will have passed

but will have a note indicating that they have not been implemented. This is because you have not yet specified the testing parameters.

10. Repeat steps 1-6 above for the remaining functions, entering suitable parameters in the format:

```
$this->assertEquals($this->Calculator-
>subtract/divide/multiply(x, y),z);
```

Select each required operation (subtract, divide or multiply), and enter variables where x and y are the two parameters which will be entered into the calculator, and z is the expected result.

11. Run the Unit Test again by clicking the Run Last Test button in the PHPUnit view and check that all the tests have passed successfully.

Notes:

1. Tests can be written in alternate ways. So for example

```
$this->assertEquals($this->Calculator->add(1, 2), 3);
```

can also be written as:

```
$this->assertSame(3,$this->Calculator->add(1,2));
```

2. You can create more than one test for each function, so that your function could appear as the following:

```
Tests Calculator->add()
 */
public function test_add()
{
$this->assertEquals($this->Calculator->add(1, 2), 3);
// $this->assertEquals($this->Calculator->add(1, 3), 4);
// $this->assertEquals($this->Calculator->add(-1, 2), 1);
}
```

Analyzing Errors

Once a PHPUnit test has been run, the results can be viewed and analyzed in order to diagnose and correct problematic sections of code.

The following steps demonstrate how to analyze and correct errors in your code:

1. To simulate a failed result, change the parameters under the add function so that the expected result is wrong. For example:
   ```
   $this->assertEquals($this->Calculator->add(1, 2),4);
   ```
2. Save the file.
3. Run the Unit Test again by clicking the Run Last Test button in the PHPUnit view.
4. The display in the PHPUnit view will now display that test_add has failed, indicated by the blue X icon .
   The error message "Failed asserting that <integer:3> is identical to <integer:4>" indicates that the test failed as the actual result of the test was 3, but the expected result was 4.
5. To only view the failures, click the "Show failures only" button on the view's toolbar.
6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon to filter the results and view the relevant functions.
7. Double-click the failed result to go to the relevant section in the code.
8. Correct the code, save the file and run the test again by clicking the Run Last Test button in the PHPUnit view.

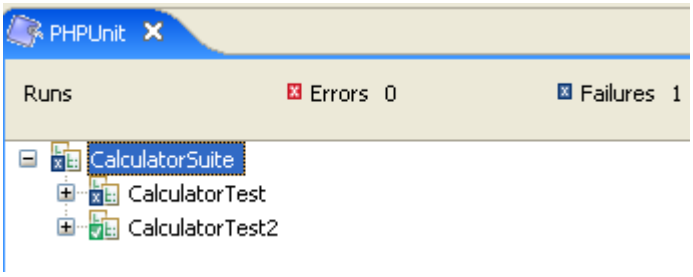
The tests should be successful. If they are not, repeat steps 6-8.

Creating and Running a PHPUnit Test Suite

A number of different PHPUnit Test Cases can be unified into one UnitTest Suite file which will run all unit tests at once. This function is useful if you have a few tests within a project which you would like to run at once.

The following steps demonstrate how to create a PHPUnit Test Suite:

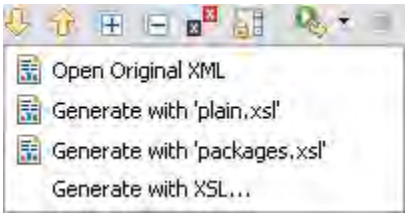
1. Create another Unit Test Case for your "Calculator" file by following steps 3 - 7 under "Creating Unit Test Cases", above.
2. Edit the test file to create different tests using your own tests, or copy-paste the example code into the file.
   (See the "Working with PHPUnit Testing"" Tutorial in Zend Studio's Online Help for the example code.)
3. Save the file.
4. From PHP Explorer View, select and right-click the Calculator project.
5. Select **New | Other | PHP | PHPUnit | PHPUnit Test Suite**.
   A New PHPUnit Test Suite dialog will open.
6. Ensure that both your test cases are selected from the list.
7. Click **Finish**.
8. A new CalculatorSuite file will be created, integrating both tests into one file.
9. Run the CalculatorSuite by clicking the arrow next to the Run button on the toolbar and select **Run As | PHPUnit Test** –or- go to Run Menu and select **Run As | PHPUnit Test** .
10. Both tests will be run, with the results of both displayed in a tree in the PHPUnit view at the bottom of the screen.



Generating PHPUnit Test Reports

The following steps demonstrate how to run a report on your Unit Test results:

1. Once you have run the PHPUnit Test Suite, click the arrow next to the Generate Report icon on the PHPUnit view's toolbar and select **Generate with 'plain.xsl'** from the drop-down list.
   See PHPUnit Testing for more on the different types of reports.



A report will be automatically generated and opened in a browser window.

Working with Zend Server

The purpose of this tutorial is to teach you how to deploy, profile and debug files and applications using the Zend Server

Contents:

Purpose and Usage
Configuring Zend Server Integration
Creating a project on a Local Zend Server
Diagnosing and Profiling Performance Issues on Zend Server
Diagnosing and Debugging Errors On Zend Server

Purpose and Usage

This tutorial demonstrates how to use Zend Studio's integration with Zend Server in order to constantly monitor your application and easily detect and diagnose performance issues and code errors during run-time.

Note:

You must have Zend Server installed on your local machine before commencing this tutorial. See Zend Server Integration for more information on the Zend Server and where to download.

Configuring Zend Server Integration

A Zend Server installed on the same machine as Zend Studio is automatically detected and configured in Zend Studio.

To automatically configure a Zend Server:

The auto detection is triggered when Zend Studio is launched or when the Auto Detect Zend Server button is clicked.

For auto detection when Zend Studio is launched:

1. Ensure Zend Server is installed and running on the local machine.
2. Open Zend Studio.
   A popup balloon will appear in the bottom-right corner of the window indicating that a Zend Server installation has been detected and configured.

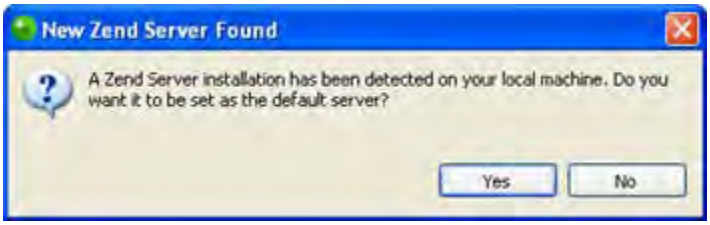
Zend Server Found
A Zend Server installation has been detected on your local machine and added as your default PHP server.
Click here to learn more

3. Click the icon to close the balloon or the 'click here' link to be taken to the Zend Server Integration help page.

To auto detect Zend Server after Zend Studio has been launched

You can manually trigger a Zend Server auto detection through the Servers view by performing the following steps:

1. Open the Servers view.
   This is available from the default PHP Perspective or can be manually opened by going to **Window | Show View | Zend Servers | Servers**.
2. Click the Auto Detect Zend Server icon on the toolbar.
3. If a Zend Server installation has been detected, a New Zend Server Found dialog is displayed asking whether you want to configure it to be the default server.


New Zend Server Found
A Zend Server installation has been detected on your local machine. Do you want it to be set as the default server?
Yes
No

4. Click **Yes** to set it as the default server or **No** for the server to be configured by but not set as the default.

A Local Zend Server configuration is configured and added to your PHP Servers Preferences page.

Creating a project on a Local Zend Server

Through Zend Studio, you can create new PHP projects directly on your local Zend Server's document root.

Note:
Ensure you have configured your local Zend Server integration, as described above.

To create a new PHP project on Zend Server;

1. From the menu bar, go to **File | New | PHP Project**.
   The New PHP Project dialog is displayed.
2. Enter 'ZendServerProfiling' in the project name field.
3. Under the contents category, ensure the 'create project on a local server' option is selected.
   The directory path should point to the document root of your Zend Server.
4. Click **Finish**.
   The project will be created on your Zend Server's document root.

Diagnosing and Profiling Performance Issues on Zend Server

Zend Server detects performance issues in your code and stores them as events. These events can then be profiled to diagnose the cause of the slow performance.

This procedure creates a purposeful performance issue in order to demonstrates how to profile an event:

1. Create a new PHP file in the project you created above by right-clicking the project in PHP Explorer view and selecting **New | PHP File**.
2. Name the file c-to-f_convert.php and click **Finish**.
3. Copy/paste the example code into the file.
   (See the "Working with Zend Server" Tutorial in Zend Studio's Online Help for the example code.)
4. Save the file.
5. Run the application you created by right-clicking the c-to-f_convert.php file in PHP Explorer view and selecting **Run As | PHP Web Page**.
   The Run PHP Web Page dialog is displayed.
6. Ensure the URL in the Launch URL field points to the location of your file on Zend

Server and click **OK**.

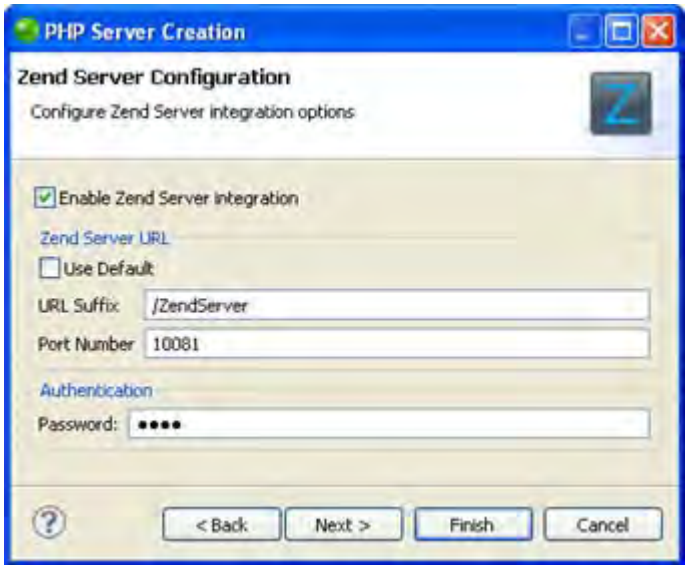
The application is launched in Zend Studio's browser.

7. In the temperature field, enter '30' and click the **Convert!** button.
   Because the code used for this application contained a purposeful slowdown, this action should have been detected by Zend Server and saved as a performance event.
8. Open the Zend Server Event list by right-clicking your Zend Server configuration in the Servers view and selecting Show Server Event List

   Or by clicking the arrow next to the Zend Server icon on the toolbar and selecting your Zend Server configuration from the drop-down list.

   If you have not yet configured your Zend Server password:

   i. A message will appear asking whether you want to configure this now.
   ii. Click **Yes** to be taken to the Zend Server preferences tab.



iii. In the Authentication field, enter your Zend Server UI password and click **Finish**.

iv. Reopen the Zend Server Event List as described in Step 8 above.

9. The Zend Server Event list is displayed.

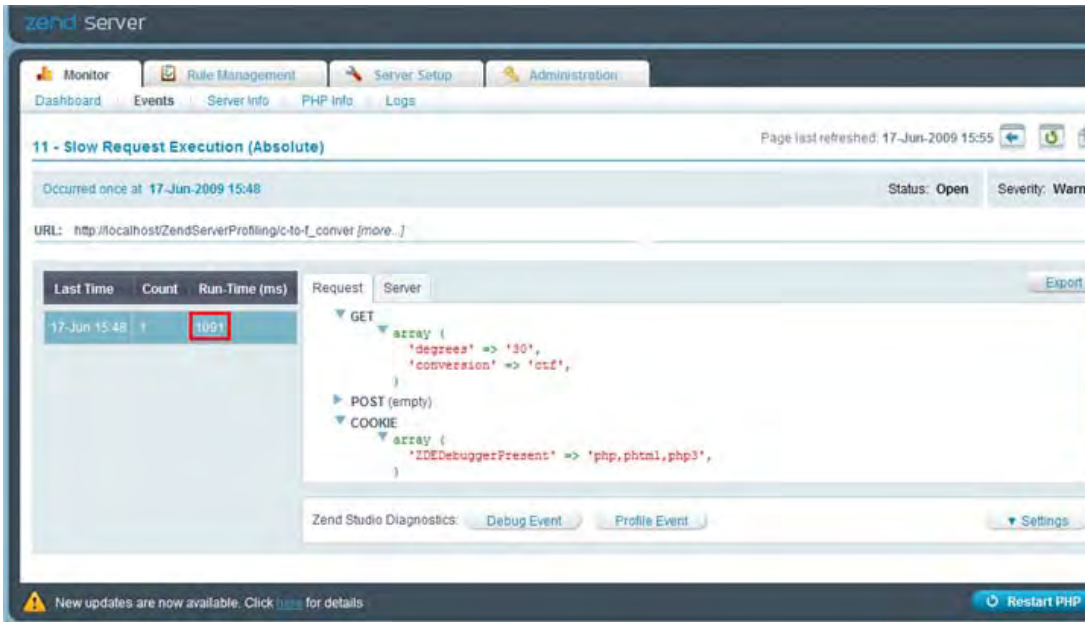
   The last event displayed should be a Slow Request Execution (Absolute) Event created by our code slow down.

10. Click the event to see the event details.

The event run time was higher than the one defined in the monitoring rules threshold.

Note:

To change this threshold, go to the Rule Management tab and click **Edit** for the Severe Slow Request Execution (Absolute) rule.



11. Click **Profile Event**.

    A profiling session is launched in Zend Studio using the same parameters as were used during the slow request execution.

Note:

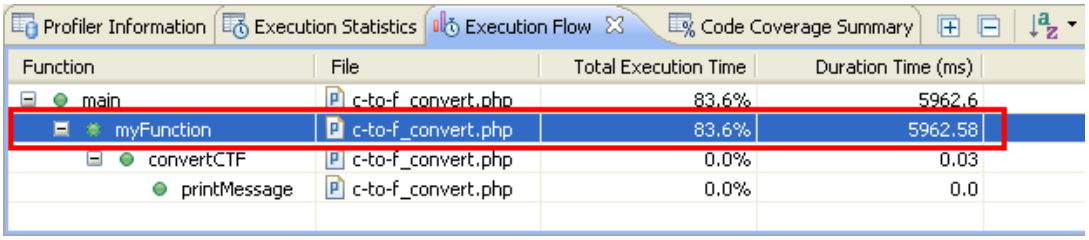
If no session is launched, see Setting Up Zend Server Integration to ensure you have correctly configured your integration settings in both Zend Studio and Zend Server.

12. Click **Yes** if asked to open the PHP Profile perspective.

    The Profiling views will display information regarding the request.

13. Explore the Profiling Execution Flow to detect the execution bottle-neck.

    You should see that the execution of ’r;myFunction ’r; took most of the time.



14. Double-click **myFunction** to view the code in the editor.

Now that the root-cause of the problem has been detected, you can debug or browse your code to diagnose and fix the error.

Diagnosing and Debugging Errors On Zend Server

Zend Server detects PHP errors in your code during run-time and stores the information as events. These events can then be debugged to diagnose the cause of the error.

This procedure creates a purposeful PHP error event in order to demonstrates how to debug and diagnose the cause of the error:

1. In the same project as was created above, create a new PHP file by right-clicking the project in PHP Explorer view and selecting **New | PHP File**.
2. Name the c-to-f_convert2.php and click **Finish**.
3. Copy/paste the example code into the file.
   (See the "Working with Zend Server" Tutorial in Zend Studio's Online Help for the example code.)
4. Save the file.
5. Run the application you created by right-clicking the c-to-f_convert2.php file in PHP Explorer view and selecting **Run As | PHP Web Page**.
   The Run PHP Web Page dialog is displayed.
6. Ensure the URL in the Launch URL field points to the location of your file on Zend Server and click **OK**.
   The application is launched in Zend Studio's browser.

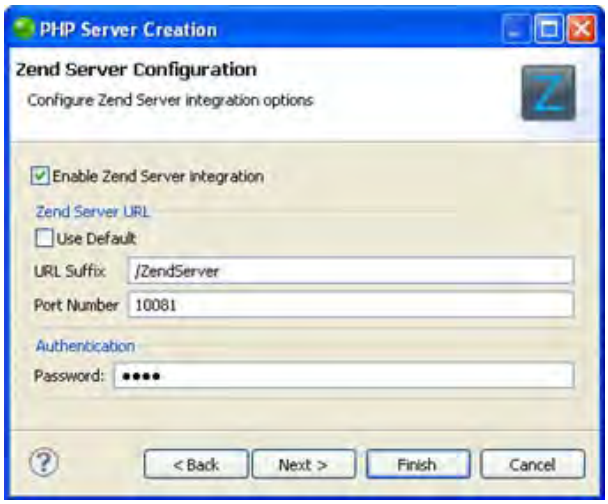
7. In the temperature field, enter '30' and select the Fahrenheit to Celsius conversion type in the drop-down list.
8. Click the **Convert!** button.
9. Open the Zend Server Event list by right-clicking your Zend Server configuration in the Servers view and selecting **Show Server Event List**.

   -Or- by clicking the arrow next to the Zend Server icon on the toolbar and selecting your Zend Server configuration from the drop-down list.

If you have not configured your Zend Server password:

i. A message will appear asking whether you want to configure this now.

ii. Click **Yes** to be taken to the Zend Server preferences tab.

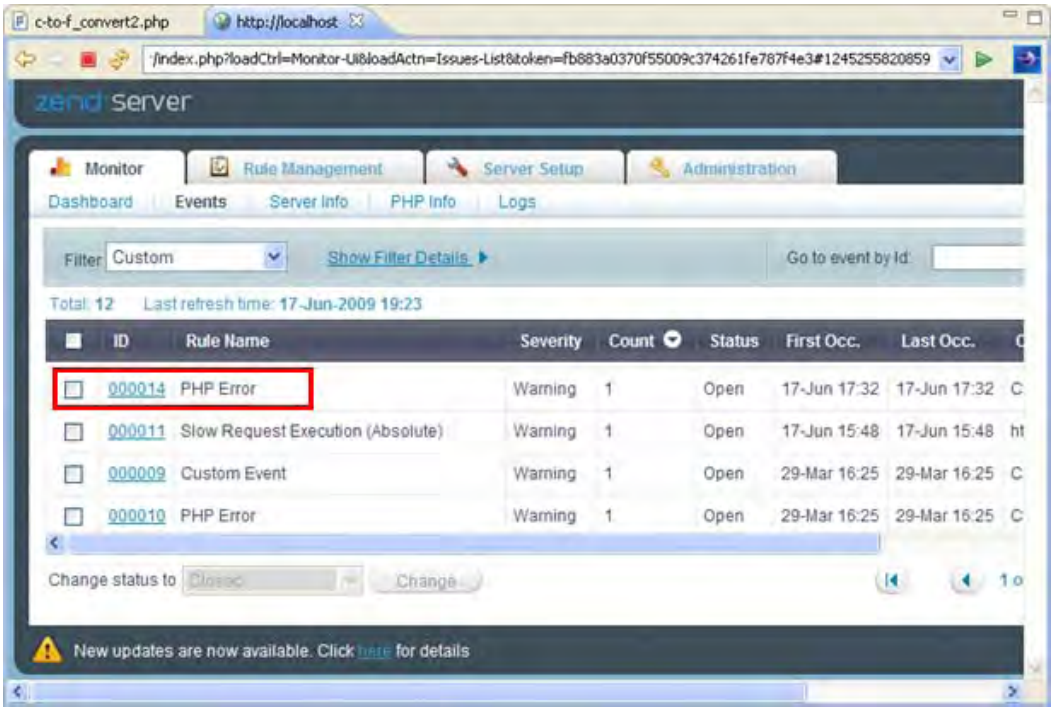


iii. In the Authentication field, enter your Zend Server UI password and click **Finish**.

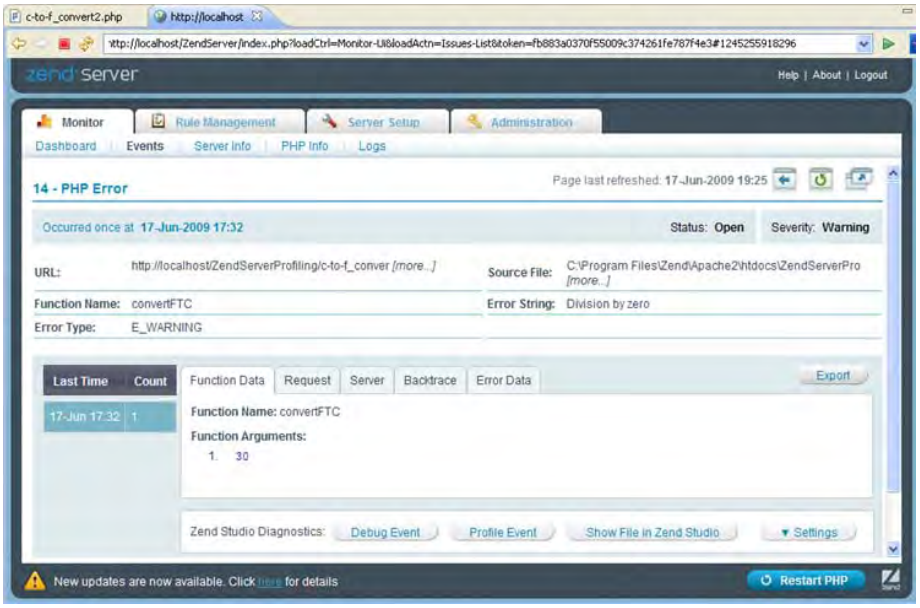
iv. Reopen the Zend Server Event List as described in Step 9 above.

10. The Zend Server Event list is displayed.

The last event displayed should be a PHP Error as a result of our PHP code error.


c-to-f_convert2.php
http://localhost
zend server
Monitor
Rule Management
Server Setup
Administration
Dashboard
Events
Server Info
PHP Info
Logs
Filter Custom
Show Filter Details
Go to event by Id:
Total: 12 Last refresh time: 17-Jun-2009 19:23
ID Rule Name Severity Count Status First Occ. Last Occ.
000014 PHP Error Warning 1 Open 17-Jun 17:32 17-Jun 17:32
000011 Slow Request Execution (Absolute) Warning 1 Open 17-Jun 15:48 17-Jun 15:48
000009 Custom Event Warning 1 Open 29-Mar 16:25 29-Mar 16:25
000010 PHP Error Warning 1 Open 29-Mar 16:25 29-Mar 16:25
Change status to
Change
New updates are now available. Click here for details

11. Click the event to see the event details.



12. Browse the Function Data and Request tabs to see the event details.
13. Click **Debug Event**.

    A debugging session is launched in Zend Studio using the same parameters as were used during the initial PHP error occurrence.

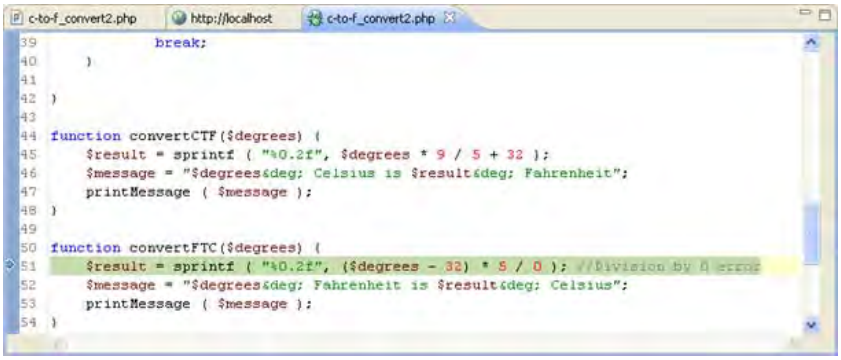
Note:

If no session is launched, see Setting Up Zend Server Integration to ensure you have correctly configured your integration settings in both Zend Studio and Zend Server.

14. Click **Yes** if asked to open the PHP Debug perspective.

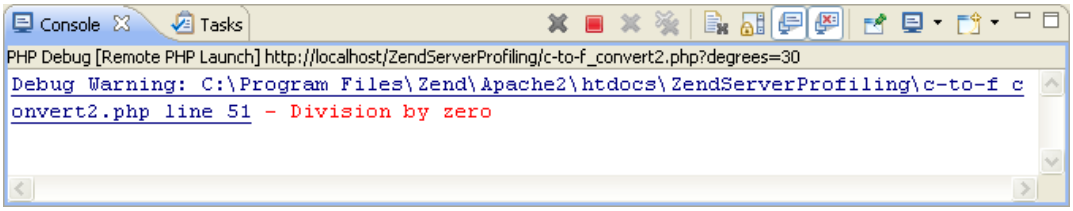
    The debug views display information about your code.

15. Click the Resume button in the Debug view or click F8 to be taken to your problematic line of code.


c-to-f_convert2.php
http://localhost
c-to-f_convert2.php
break;
function convertCTF($degrees) {
$result = sprintf ("%0.2f", $degrees * 9 / 5 + 32);
$message = "$degrees° Celsius is $result° Fahrenheit";
printMessage ($message);
function convertFTC($degrees) {
$result = sprintf ("%0.2f", ($degrees - 32) * 5 / 0); //Division by 0 error
$message = "$degrees° Fahrenheit is $result° Celsius";
printMessage ($message);

16. Click the Step Over button in the Debug view or click F6.

    A Debug Warning message appears in the Console view detailing the error and its cause.


Console
Tasks
PHP Debug [Remote PHP Launch] http://localhost/ZendServerProfiling/c-to-f_convert2.php?degrees=30
Debug Warning: C:\Program Files\Zend\Apache2\htdocs\ZendServerProfiling\c-to-f_convert2.php line 51 - Division by zero

The code can now be easily fixed (in this case by replacing the "5 / 0" operation with "5 / 9").

Concepts

Update Manager	Debugging
PHP Version Support	Profiling
Content Assist	Breakpoints
Syntax Coloring	PHP Include Path
Automatic Completion	PHP Build Path
Matching Brackets	Path Mapping
Mark Occurrences	Zend Browser Toolbar
Code Folding	Tunneling
Code Commenting | phpDoc Block Comments | Bookmarks	Zend Server
Hover Support	Code Tracing
Override Indicators	PHPUnit Testing
PHP Working Sets	Refactoring
Type Hierarchy	JavaScript Support
PHP Manual Integration	PHPDocs
Real Time Error Detection | Code Analyzer and Semantic Analysis	Code Galleries
Local History	Zend Guard Integration
CVS	RSS Feeds
SVN	WSDL - Web Services Description Language
Zend Framework Development	Zend Studio for IBM i Extras
Database Connectivity	Remote Server Explorer
Running	Mylyn Integration
	Phar Integration

Update Manager

Zend Studio's Update Manager allows for the easy installation of extra plug-ins, the updating of existing features and the easy updating of Zend Studio with the latest offerings from Zend.
See the Workbench User Guide for more on installing and updating features with the Update Manager.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

PHP Version Support

Zend Studio supports PHP versions 4, 5.1/5.2 and 5.3.

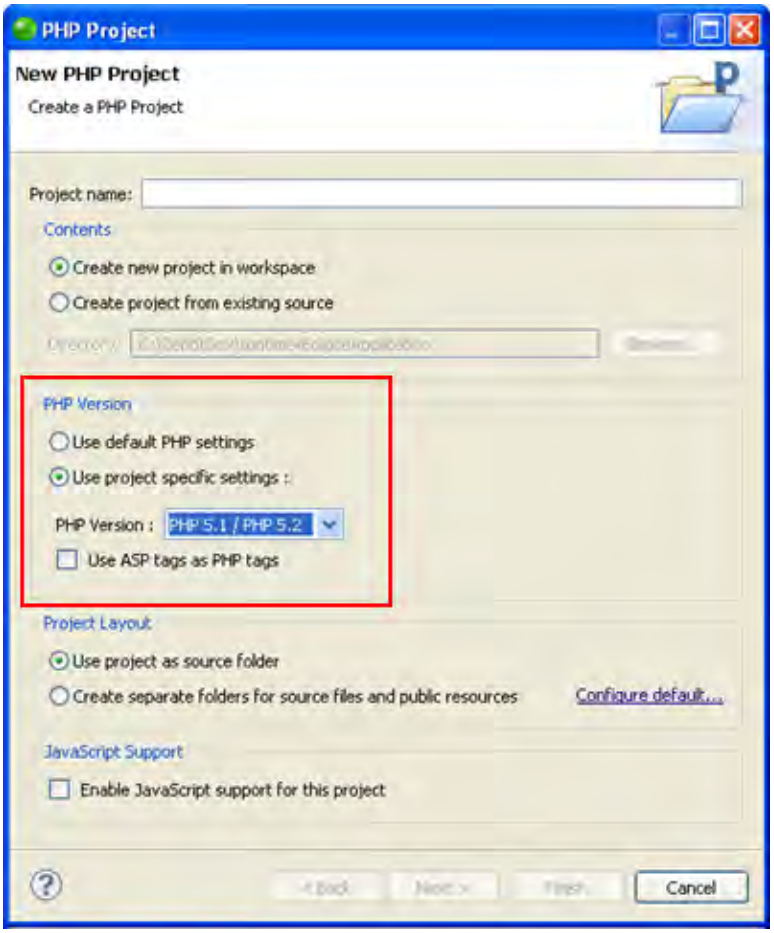
PHP version settings affect:

- The elements displayed in the PHP Functions view.
- The options available in Content Assist.
- Debugging and Profiling functionality.

PHP version settings can be configured from the following places:

- PHP Executables can be added and edited from the PHP Executables Preferences page.
- The default PHP executable used for new projects can be set in the PHP Interpreter Preferences page. Through this page you can also set the PHP version for specific projects.

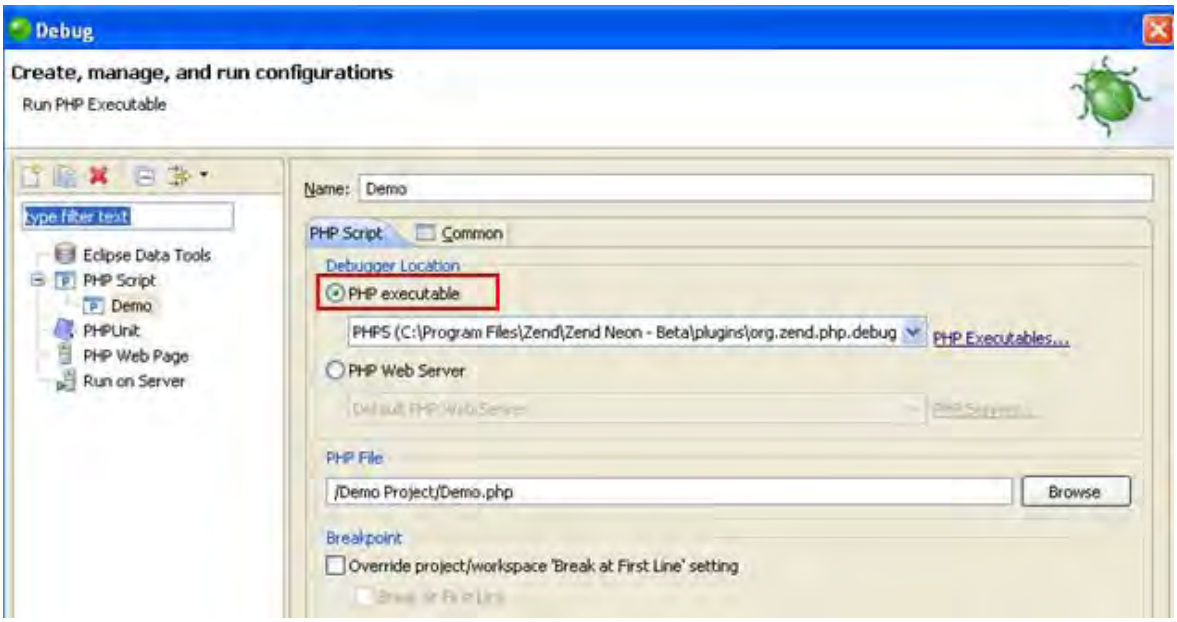
  In addition, you can select which PHP Version should be used when creating a new PHP project by marking the Enable Project Settings checkbox in the New PHP Project dialog.



- The default PHP executable used with the debugger can be set in the Debugging Preferences page, accessed from **Window | Preferences | PHP | Debug**. Through this page you can also set the PHP executable used to debug specific projects.
  In addition, you can also configure the PHP executable used for each Debug and Profile session in the Debug/Profile configuration screens.

Note:

In order to perform local debugging for PHP 4.x projects, you must manually add the relevant PHP executable to the PHP Executables Preferences page.



Note:

In order to minimize errors, the PHP Executable used for debugging/profiling should match the PHP version used for the project.

Content Assist

The Content Assist feature enables the selection and insertion of existing code elements to complete partially entered code.

A list of possible code elements is displayed in relevant locations within your files according to the context of your cursor, which you can then select to be automatically entered into your code.

Each type of code element will have a unique icon. See PHP Icons for a description of each of the icons.

- Reserved PHP Words
- Functions
- Templates
- Classes
- Interfaces
- Constants
- Variables (public)
- PHP File Include Call
- Namespaces (PHP 5.3)

Content Assist works with the following elements: PHP Classes, Functions, Variables, Constants, Keywords, Interfaces, attributes, values, nested functions, names, syntax and include calls, as well as all user defined Classes, Functions and Constants.

Note:

Content Assist will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

Using elements within the same scope

Elements within the same active project, file or function will be available for use with Content Assist.

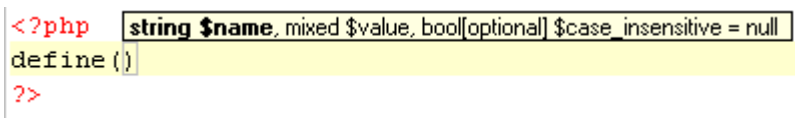
Examples:

- Variables within a function will be added to the Content Assist list when the cursor is within that function.
- Elements defined within one file will be available in the Content Assist menu in other files within the same project.

Function Parameter Hints

When entering a function call, a Function Parameter Hint box will be displayed detailing the types of parameters which should be entered within the parentheses of the function call.

Example:

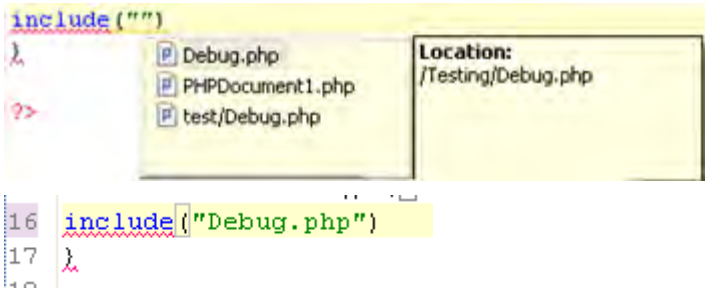


If the Function Parameter Hint box is not displayed automatically, place your cursor between the parentheses and press **Ctrl+Shift+Space**.

Content Assist for Include Statements

Content Assist can be activated for require and include calls to call files contained within the same project.
Inserting quotation marks between the parentheses of an include/require call will cause the Content Assist window to display the files available for the function. Selecting a file will complete the include/require call.

Example:



Class Type Hints

By using a comment you can assign a variable its exact class value. This assignment will affect the content assist of this variable accordingly.

Example:

```
<?php

function getClass() {

    return new Test ( );

}
class Test {
    function printValues( $a,  $b) {
        echo "Values: $a, $b";
    }
}
/* @var $myVar Test */
$myVar = getClass ();
$myVar->
```

printValues($a, $b) - Test

Location
PHPFiveThree\abc.php
Class
Test

Note:

Without the comment, content assist will not be available for the function.

By using a comment you can assign a variable its exact class value. This assignment will affect the content assist of this variable accordingly.

To assign a variable its class value using a comment:

1. Create your function and assign variables to it.
2. Enter a comment in the form:

```
/* @var $"new variable" "Class name" */
```

Content Assist for Magic Members

Zend Studio supports Content Assist options for 'magic members'. These are properties and methods which were declared using the @property or @method tags within PHP DocBlock comments rather than in the PHP script itself.

See http://manual.phpdoc.org/HTMLSmartyConverter/PHP/phpDocumentor/tutorial_tags.property.pkg.html for more information on magic members.

Example:

```
<?php
/**
 * @property-read int $foo the foo prop
 * @property-write Magician $bar the bar prop
 * @method int borp() borp(int $int1, int $int2) multiply two integers
 */
class Magician {
    private $_thingy;
    private $_bar;
}

$blah=new Magician();

$blah->
?>
```

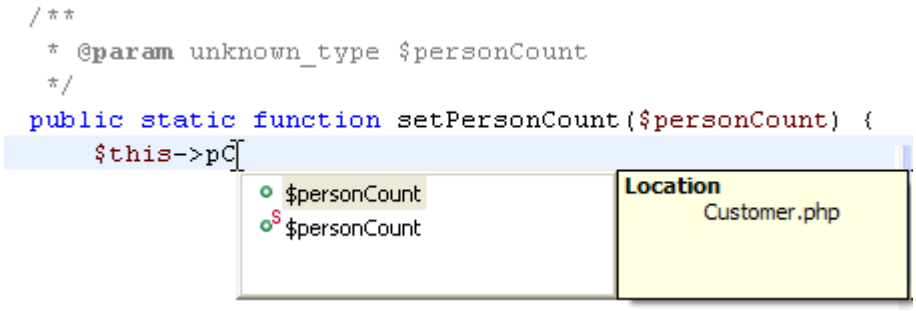
borp()
bar
foo

Location
MagicMembers.php
Class
Magician

Camel Case Matches

Content Assist supports camel case matches when entering code, which displays Content Assist options based on an element's initials.

Example:



Note:

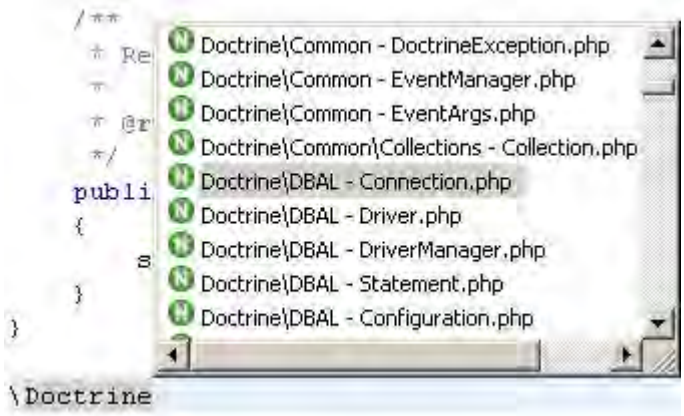
Camel Case matching is case sensitive.

Namespaces

When the PHP executable used for the project is set to version 5.3, content assist is available for namespaces.

Namespace elements are represented by the Namespace icon .

Example:



Configuring Content Assist

To configure PHP Content Assist options, go to the Content Assist Preferences page, accessible from **Window | Preferences | PHP | Editor | Content Assist**.

To configure JavaScript Content Assist options, go to the JavaScript Content Assist Preferences page, accessible from **Window | Preferences | Web | JavaScript | Editor | Content Assist**.

Syntax Coloring

Zend Studio can automatically apply different colors and font attributes to different PHP syntax elements in order for your scripts to be more easily navigable and to help you find relevant sections quickly and easily.
Completing the relevant element will cause the required color and font settings to be applied to it.

The default color and font settings for PHP elements are:

- PHP tags
- Comment
- HereDoc
- Keyword
- Normal
- Number
- PHPDoc
- String
- Task Tag
- Variable

The color and font settings can be configured from the Syntax Coloring preferences page, accessed from **Window | Preferences | PHP | Editor | Syntax Coloring**.

Note:

Syntax Coloring will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.
To configure JavaScript syntax coloring preferences, go to **Window | Preferences | JavaScript | Editor | Syntax Coloring**.

Automatic Completion

Zend Studio can be set to automatically complete certain types of patterns. To use the auto-complete function, type the opening character of the pattern in the editor and press **Enter**. The pattern will be automatically completed by the relevant characters being inserted.

The following types of patterns can be auto-completed:

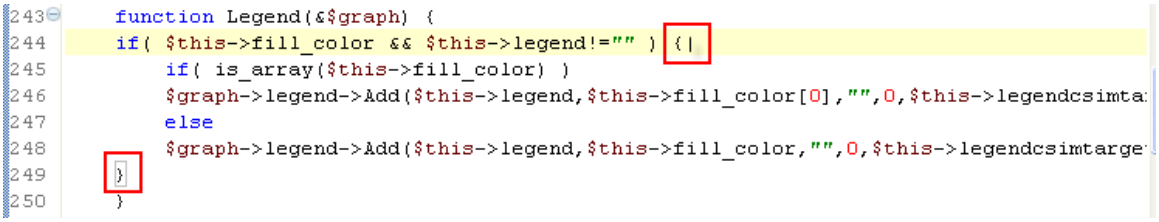
- "Strings" - Automatically inserts a pair of double quotes.
- (Parentheses) and [Square] brackets - Automatically inserts a pair of brackets.
- {Braces} - Automatically inserts a pair of curly brackets
- PHPDoc and comment regions - Automatically inserts the "/** **\ " PHPDoc characters.

The types of patterns that can be auto-completed can be configured from the Typing Preferences page, accessible by going to **Window | Preferences | PHP | Editor | Typing**.

Matching Brackets

Zend Studio can help you to easily navigate through your script by finding brackets' matching pairs.

To see a bracket's pair, click to the right of the bracket. Its matching pair will be highlighted.
To jump to the matching bracket, press **Ctrl+Shift+P**.

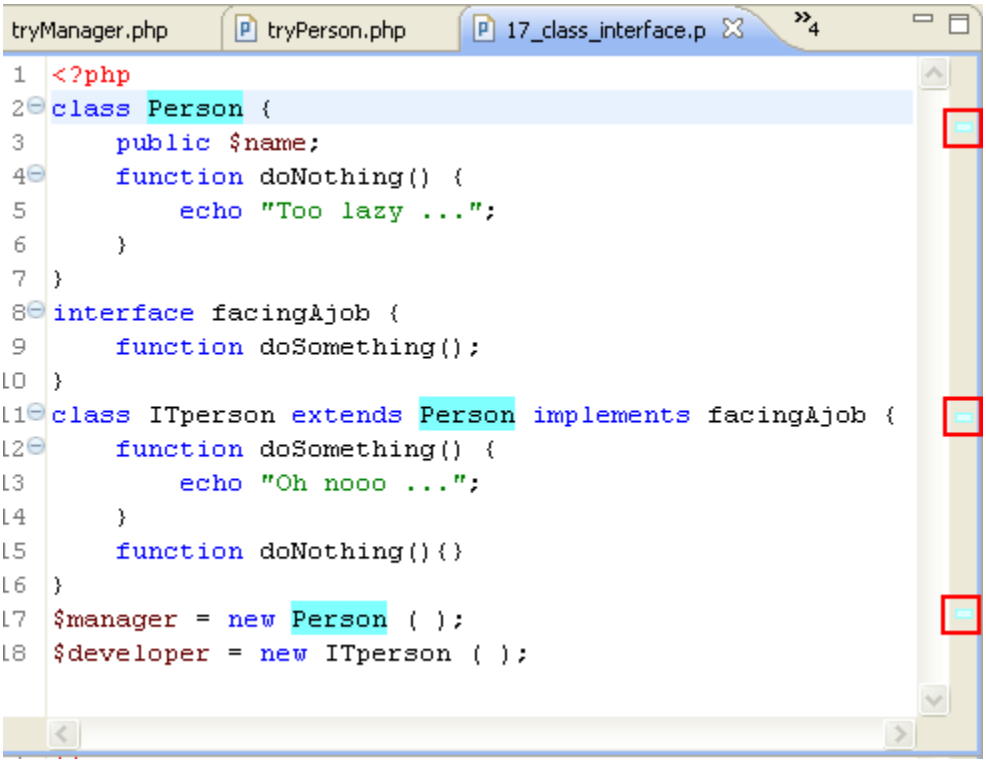


Mark Occurrences

The Mark Occurrences feature enables you to see where an element is referenced by simply clicking on the element in the editor.

When the Mark Occurrences feature is enabled, all occurrences of the element within the active file will be highlighted, and indicators will be displayed in the annotations bar (to the right of the editor).

In addition, hovering over the element will cause a tooltip to be displayed with the location of all references of the element, including occurrences of the element in other files.



Occurrences can be of 2 types - “Write Occurrence” (for occurrences that are in a write mode) or “Read Occurrence” (for occurrences that are in a read mode). These will be highlighted in different colors according to the setting configured for PHP elements 'read' and 'write' occurrences in the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Preferences**). Here you can also configure the indicators used in the vertical ruler/marker bar (to the left of the editor) and the annnotations bar (to the right of the editor).

The types of elements whose occurrences will be highlighted can be configured in the Mark Occurrences preferences page (**Window | Preferences | PHP | Editor | Mark Occurrences**).

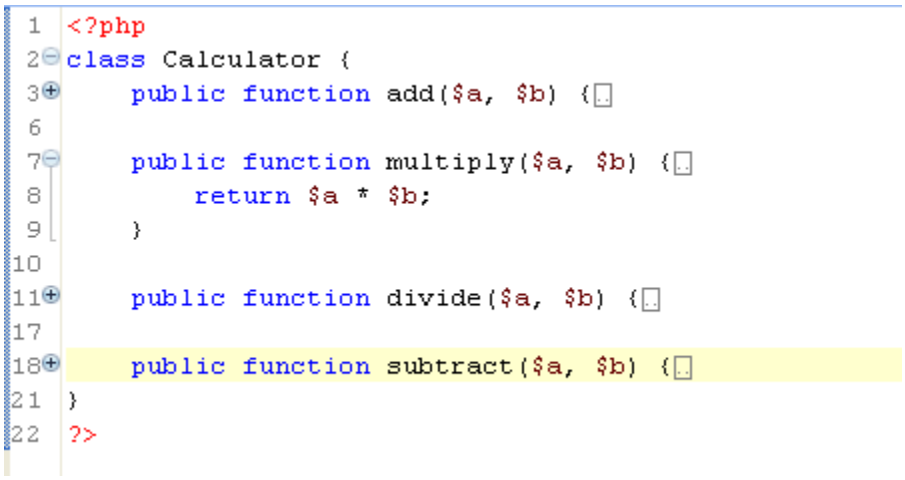
By default, the following types of elements are marked:

- Method Exit - Exit points (throws / return / end of flow) of a method will be marked
- Break/Continue - Scope (for, foreach, while, do-while or switch structure) of a break/continue statement will be marked
- Local variables
- Global variables
- Parameters
- Functions
- Methods
- Fields
- Class Constants
- Constants (defined)
- Class/Interface
- HTML Tags

Code Folding

Code Folding collapses or "folds" the display of a block of code. The editor will then display only the first line of code instead of the entire block. This reduces visual clutter by enabling users to selectively hide and display complicated text while still viewing those subsections of the text that are relevant.

Cold Folding is applied by default for functions and PHPDocBlocks. You can configure which of these are folded by default through the Folding preferences dialog.



Folded code has a plus sign [+] on the vertical marker bar to the left of the Editor. Clicking this sign will expand the block of code.

Opened, unfolded code has a minus sign [-] on the vertical marker bar to the left of the Editor. Clicking this sign will fold the block of code.

See the Using Code Folding topic for more information.

Code Commenting

Commenting your code involves adding characters (normally slashes and stars) which mark certain areas of code as 'comments'.
Comments are used for reference information only and will not be run as part of your code. It is good programming practice to comment all functions, classes and methods in your code. This helps both the developer and others who might look at the code to understand its purpose.

Comments can be either single line or multi-lined:

- Single lined comments will have the following format:

```
// This is a single line comment.
```

- Multi-lined comments will have the following format:

```
/* This is
a comment
on multiple
lines
*/
```

Note:

Code Commenting will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

phpDoc Block Comments

Zend Studio offers a preset means for adding phpDoc comments to files by providing an input line when including statements, classes, class variables, and constants to the code. Developers are prompted to immediately add a description ensuring that the added elements are documented in their context and in real-time.

phpDoc blocks are descriptive comments that are part of the application code. They are used to describe the PHP element in the exact location in the code where the element appears. The block consists of a short description, long description, and phpDoc tags.

Example:

When creating a phpDoc Block commen for the following function:

```
function add ($a, $b) {
    return $a + $b;
}
```

the following comment will be created:

```
/**
 * Enter description here...
 *
 * @param unknown_type $a
 * @param unknown_type $b
 * @return unknown
 */
```

The comments should now be edited with the relevant description and parameters.

Descriptions that are added for a code element are also automatically added to the Content Assist bank so that the next time the code element is used it is readily available from the Content Assist list. The element's descriptions will also appear in the Outline view.

Note:

Zend Studio offers Content Assist support for magic members declared in code comments. See the Content Assist concept for more information.

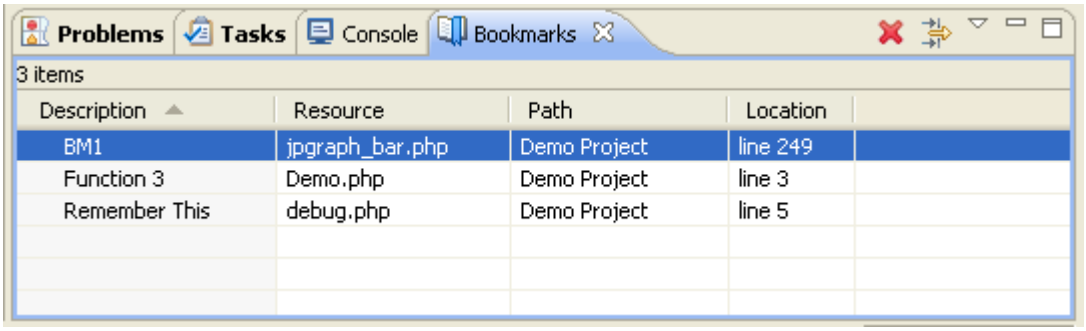
phpDoc blocks also serve as the input for creating a PHPDoc.

Bookmarks

Bookmarks can be used as placeholders within your scripts to allow easy navigation to pre-defined places within your scripts and resources.

Bookmarks are indicated by a bookmark icon in the vertical marker bar to the left of the editor.

Bookmarks are displayed in the Bookmarks view, which can be opened by going to **Window | Show View | Bookmarks**.



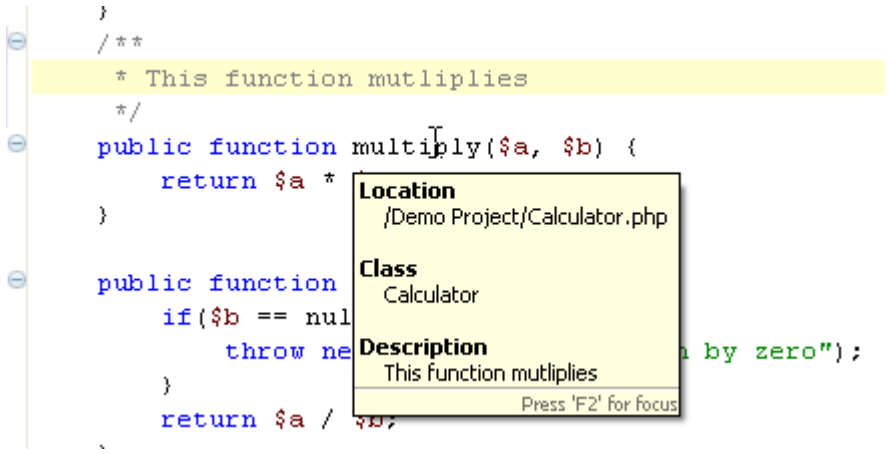
For more on bookmarks, see the Bookmarks topic in the Workbench User Guide.

Note:

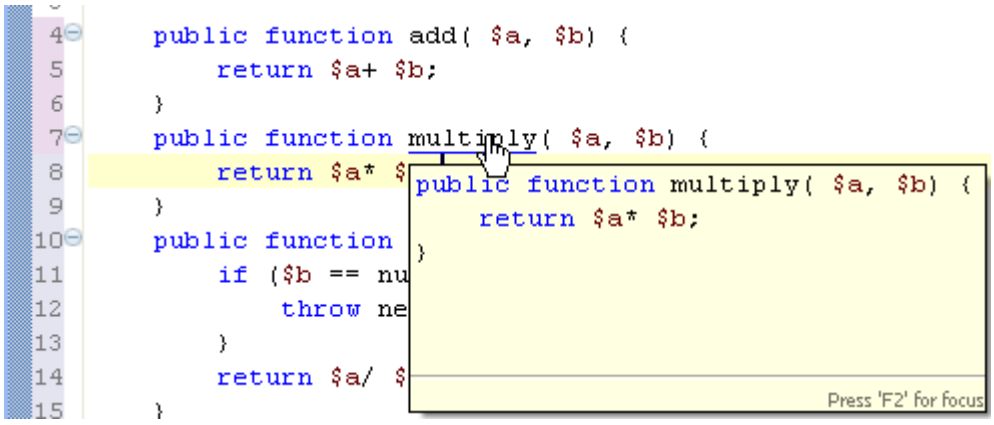
Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Hover Support

Hovering over an element will cause a tooltip to appear with information about that element, containing the location of its declaration and any additional information (description, parameters etc.) contained in the element's relevant PHPdoc comment:



Holding down **Ctrl** while hovering over an element will also show you everything contained within that element:



When hovering, press **F2** for the Hover tooltip to come into focus. This ensures that it is displayed even when not hovering and enables you to select the text from within it.

Clicking on an element while hovering will take you to that element's declaration. See Using Smart Goto Source for more information.

Configuring Hover Preferences

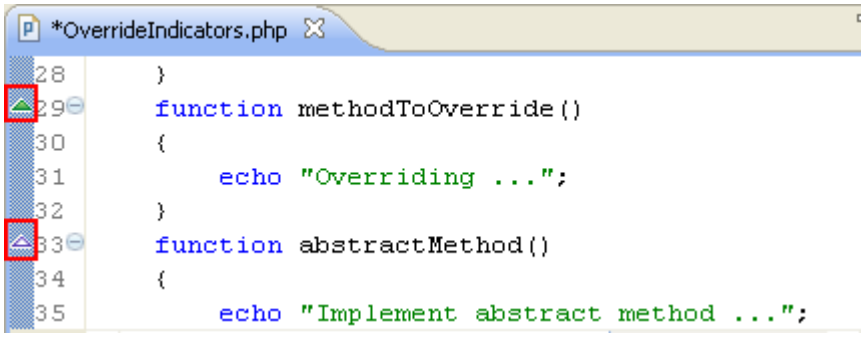
To configure your hover settings, go to the Hover Preferences page, accessible by going to **Window | Preferences | PHP | Editor | Hovers**.

Note:

Hovering will also be enabled for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information. JavaScript hover settings can be configured from the JavaScript Hovers preferences page, accessible by going to **Window | Preferences | Web | JavaScript | Editor | Hovers**.

Override Indicators

'Override Indicators' are markers that display special decoration icons to indicate methods that override or implement other methods from a super-type's class/interface.
The override indicators will be displayed as triangles in the marker bar (the vertical ruler to the left of the editor) next to the method's definition.



Green triangles indicate an overridden method, while a white triangle indicates an implemented method.

Hovering over a triangle displays a tool-tip that reveals the overridden or implemented method's super-type. Clicking the triangle will open the type in an editor.

Note:
Markers will not be displayed if the methods do not comply with the overriding rules.
These are as follows:

- You cannot override a non-static method with a static method.
- The overriding method must not be more restrictive than the overridden method
- 'Private' methods are not inherited and cannot be overridden.

Override Indicators Preferences

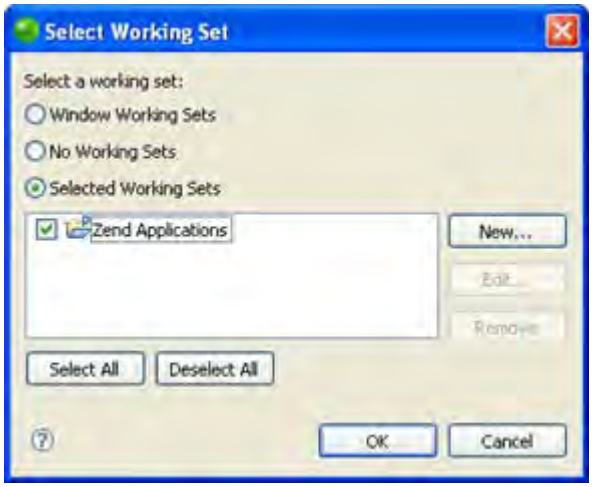
The display preferences for the override indicators can be configured from the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Annotations**). Select the Override Indicators (org.eclipse.php.ui.overrideIndicator) option to configure the Override Indicators for PHP.

PHP Working Sets

PHP Working Sets are groups of PHP resources which allow you to view or apply actions to a group of pre-defined elements. Workings Sets group elements under one title so that they can be easily selected when viewing resources or carrying out actions. Elements included in a PHP Working Set can include any PHP project, folder or file.

Example for using Working Sets:

Selecting the 'Select Working Set' option in the PHP Explorer view's context menu allows you to choose specific Working Sets (i.e. group(s) of projects/files) to be displayed in the view.



Projects/files not in the selected Working Set will not be displayed in the view.

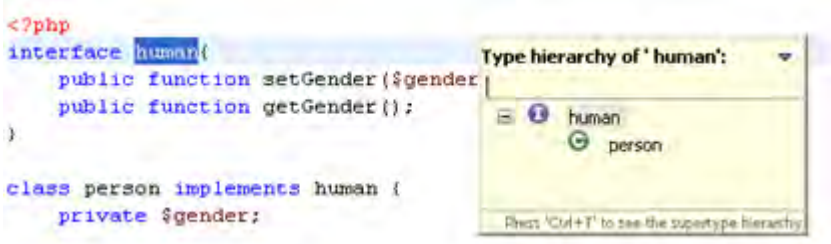
If you have not created PHP Working Sets, see Creating PHP Working Sets for more information.

Type Hierarchy

The Type Hierarchy views display the hierarchy for a given type (a class name, interface name or class methods, constants and fields.). This allows you to view an element's supertypes (types higher in the hierarchy) or subtypes (lower in the hierarchy) within a tree structure, providing you with an overview of your element's structure.

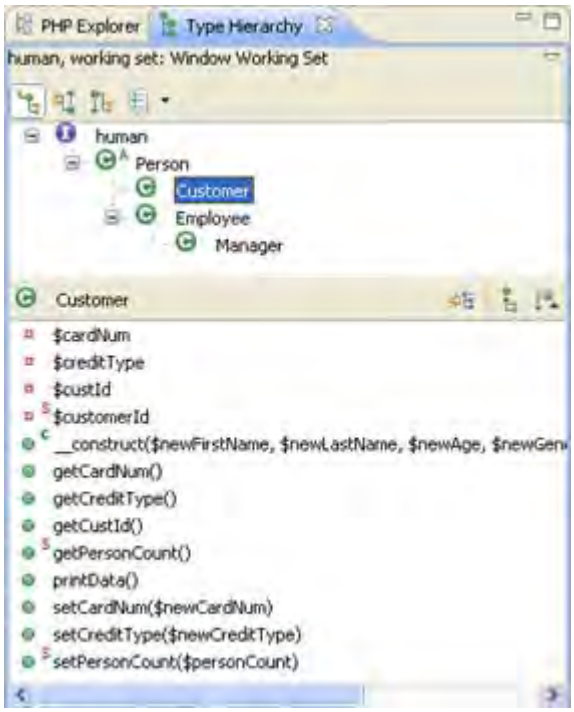
A type hierarchy can be displayed in two ways:

1. **Quick Type Hierarchy view -** Displays a simple hierarchy view within the editor.



2. **Type Hierarchy view -** A standalone view.

- The Type Hierarchy Tree - Displays the type's supertypes and/or subtypes.
- Member list pane - Displays the type's members.

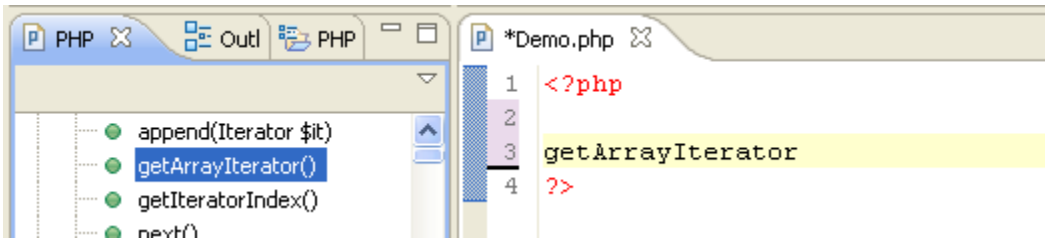


See Type Hierarchy View for more information.

PHP Manual Integration

Zend Studio can integrate with PHP Manual sites in order to get the most up-to-date PHP information.

The PHP Functions view lists all PHP functions contained within the PHP manuals, and can be used in order to easily add these functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.



You can open a PHP manual site with an explanation about most of the functions on the list by right-clicking a function in PHP Functions view and selecting **Open Manual**.

A new browser window will open with an explanation of the function from the PHP Manual site.

Note:

Some functions will not have a description assigned to them in the PHP Manual site. In this case, a browser will open with a 'Cannot find server' error message.

PHP Manual sites can be added and edited from the PHP Manual Preferences page.

In addition, The PHP Functions view can be used in order to easily add functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.

To open a PHP Manual, right-click in the Editor and click **Open PHP Manual** -or- press **Shift + F2**.

Real Time Error Detection

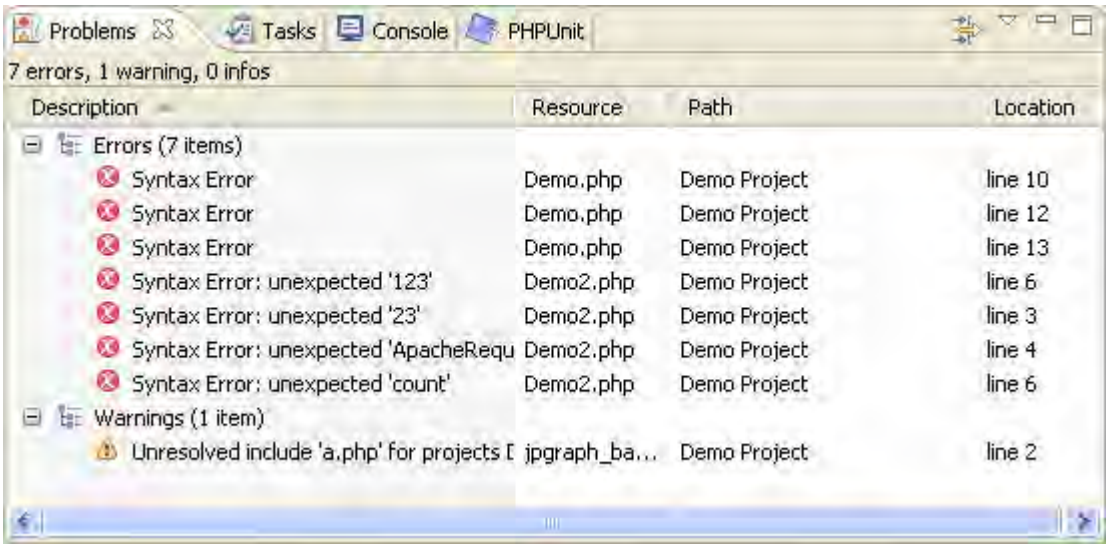
Zend Studio automatically highlights errors and problems in your PHP script.

Errors and warnings will be displayed in the Problems view, accessed from **Window | Show View | Other | General | Problems**.

In addition, error icons and warning icons will be displayed in the vertical marker bar to the left of the editor window, as well as next to the relevant project in PHP Explorer view (these will be indicated in the PHP project/file icons - e.g.).

All warnings, errors and problems in open projects will be logged in the Problems view, which displays the following information:

- Description - A detailed description of the error.
- Resource - The name of the resource containing the problem.
- Path - The name of the Project containing the resource.
- Location - The line number(s) of the error within the file.
- Type - The Type of error occurring.


Problems
Tasks
Console
PHPUnit
7 errors, 1 warning, 0 infos
Description
Resource
Path
Location
Errors (7 items)
Syntax Error Demo.php Demo Project line 10
Syntax Error Demo.php Demo Project line 12
Syntax Error Demo.php Demo Project line 13
Syntax Error: unexpected '123' Demo2.php Demo Project line 6
Syntax Error: unexpected '23' Demo2.php Demo Project line 3
Syntax Error: unexpected 'ApacheRequ Demo2.php Demo Project line 4
Syntax Error: unexpected 'count' Demo2.php Demo Project line 6
Warnings (1 item)
Unresolved include 'a.php' for projects [jpgraph_ba... Demo Project line 2

The Problems view groups problems according to Errors, Warnings or Info.

Double-clicking on an error in the Problems view will take you to the relevant location in the Editor.

If the Problems view is not displayed, go to **Window | Show View | Problems**.

Semantic Analysis

Zend Studio's Semantic Analysis mechanisms can detect problems beyond the regular parsing warnings and errors. This helps developers to analyze static source code to enforce good coding practices and scan PHP code. The Semantic Analysis feature achieves this functionality by attempting to reconcile problematic code and locating unreachable code (code that has been defined but is not used or with empty variables).

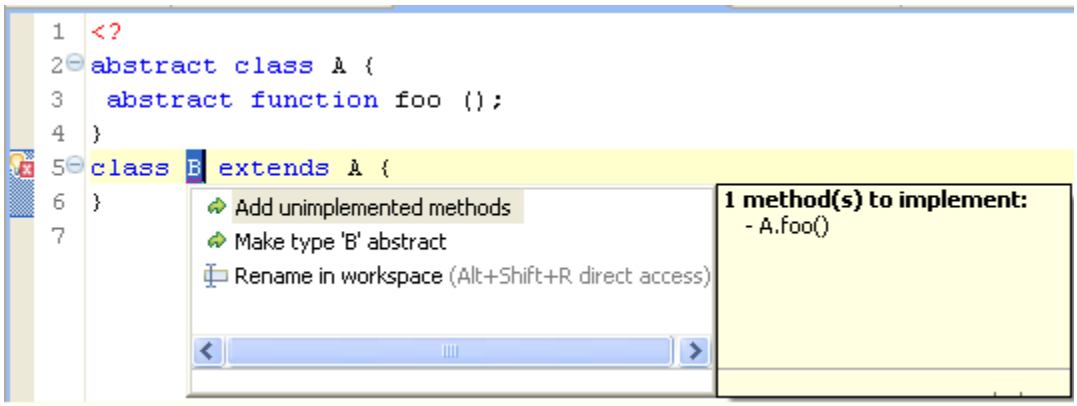
Semantic Analysis warning messages help to ensure that your code is written and formatted in a way that will result in optimal performance for your script. In addition, it supplies you with practical suggestions for improving the code.

Semantic Analysis problems will be displayed in the Problems view and be indicated in the code with warning and error icons (see Real Time Error Detection for more information).

Quick Fix

The Semantic Analysis Quick Fix option enables you to easily change your problematic code according to suggestions supplied by the Semantic Analysis mechanism. This includes suggested actions for element creation or implementation.

Example:



See Applying Quick Fixes for more information.

Preferences

To enable/disable Semantic Analysis and to configure which occurrences will trigger warning or error messages, go to the Semantic Analysis Preferences page, accessible by going to **Window | Preferences | PHP | Semantic Analysis**.

Local History

A Local History of a file is maintained when you create or modify and save a file. Each time you edit and save a new version of a file, a copy of it is saved in local history.

Each state in the local history is identified by the date and time the file was saved.

This allows you to compare your current file state to a previous state, replace the file with a previous state, or restore a file that was deleted.

Note:

There is no Local History associated with Projects or with Folders.

See the Using Local History topic for more information.

CVS

A Concurrent Versions System (CVS) repository is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to revert file and project states back to previous versions.

CVS functionality can be accessed from the CVS Repository Exploring Perspective, accessed by going to **Window | Open Perspective | Other | CVS Repository Exploring**.

See 'Working in a Team Environment with CVS' in the Workbench User Guide for more information.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

SVN

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.
SVN functionality can be accessed from the SVN Repository Exploring Perspective, accessed from **Window | Open Perspective | Other | SVN Repository Exploring**.

See the Subversive User Guide for more information on SVN.

Zend Framework Development

Zend Framework is Zend's open source, object-oriented web application framework implemented in PHP 5.

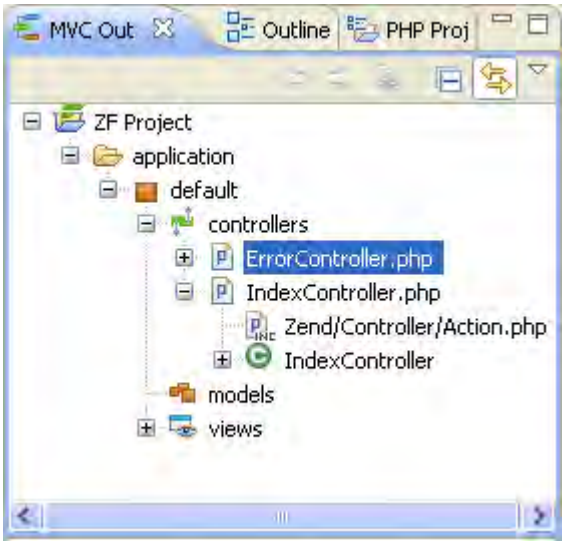
For information on Zend Framework, visit the Zend Framework site at: http://framework.zend.com or http://framework.zend.com/manual/en for the Zend Framework Reference manual.

Zend Framework Project and Element Creation

Zend Studio's Zend Framework integration functions allow you to create a new Zend Framework Project that is organized into Framework's Controller-Model-View system. The project will have Zend Framework's libraries added to its include path, allowing access to all Zend Framework's elements.

The new Zend Framework project that is created in Zend Studio will contain basic files for a simple "Hello, World!" program.

When a Zend Framework Project is created, you will be prompted to open the Zend Framework Perspective, containing the MVC Outline view.



The MVC Outline view provides an outline of all controller, model and view classes, files, variables and related functions.

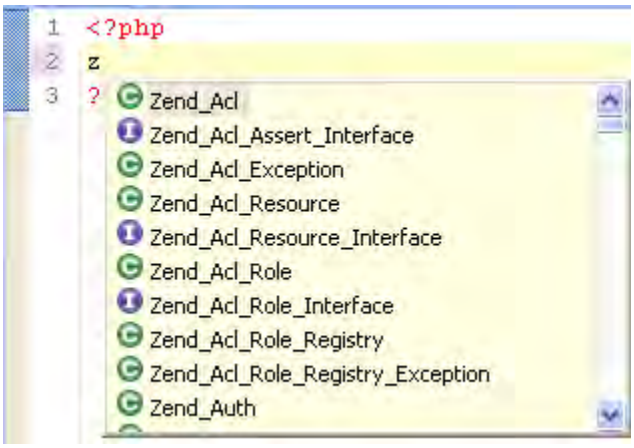
Note:

To manually open the Zend Framework Perspective, go to **Window | Open Perspective | Zend Framework**.

Once you have created a Zend Framework project, you can use Zend Studio's New Zend Table, Zend View, Zend Controller, Zend Module, Zend View Helper or Zend Action Helper Wizards to create new Zend Framework elements.

Zend Framework Content Assist

Once Zend Framework's libraries are included in a project's include path, its classes, functions, iterators and variables will be available for use from the Content Assist window. Simply press **Z** followed by **Ctrl+Space** in the Editor to view the list of available Zend Framework elements:



For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com or the Zend Framework Reference Manual at http://framework.zend.com/manual/en. For more external resources, see Useful Links.

Database Connectivity

Zend Studio's Data Tools Platform plugin allows you to connect to, view, edit, and run queries on databases. The Data Tools Platform provides connectivity with a number of databases.
Data Tools Platform functionality is accessed through the Database Development Perspective.

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Note:

Additional user guides can be accessed from inside Zend Studio by going to **Help | Help Contents**, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

For instructions on connecting to and accessing your database, see Connecting to Databases.

Running

Zend Studio allows you to run the applications you are working on from the workbench. This allows you to run and test your applications during development.

Zend Studio includes several different methods of running your files and applications:

PHP Script Local Running

Allows you to run files situated in your workspace. This enables you to locally validate freshly developed code before deploying to a Web server. Using this option, you can run applications which do not require user input or responses from the server.

Note:

When running internal files the Zend Studio internal debugger uses its own PHP executable that was installed together with Zend Studio.

See Running PHP Scripts Locally for more information.

PHP Script Remote Running

Allows you to run files situated locally on your workspace using your server's Zend Debugger. This allows you to test your files in the production environment, and allows you to utilize the extensions installed on your server.

See Running PHP Scripts Remotely for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

PHP Web Page Running

Allows you to run applications situated on a server, including any required interactive user input. The PHP Web Page Run dialog has an option to give the files you are working on first priority when running, using the "Local Copy" option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:

It's recommended that your local project structure reflect the project structure on your server.

See Running PHP Web Pages for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

Debugging

Zend Studio debugging function allows you to test your files and applications and detect errors in your code.
The debugger allows you to control the execution of your program using a variety of options including setting breakpoints, stepping through your code, and inspecting your variables and parameters.

Zend Studio includes several different debugging methods as described below:

PHP Script Local Debugging

Allows you to debug files on your workspace using Zend Studio's internal debugger.
The Internal Debugger enables developers to locally validate freshly developed code before deploying to a web server. The internal option means that files located on your workspace can be debugged. When debugging internal files the Zend Studio Internal Debugger uses its own PHP executable that was installed together with Zend Studio.

See Locally Debugging a PHP Script for more information.

PHP Script Remote Debugging

Allows you to debug files on your workspace using your server's Zend Debugger.
This allows you to debug local files using the Zend Debugger situated on your server. This allows you to test your files with the production environment, and allows you to utilize the extensions installed on your server.

See Remotely Debugging a PHP Script for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

PHP Web Page Debugging

Allows you to debug applications situated on a server. It allows you to debug whole applications, including any required interactive user input.
The PHP Web Page Debug has an option to give the files you are working on first priority when debugging, using the "Local Copy" option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:
It's recommended that your local project structure reflect the project structure on your server.

See Debugging a PHP Web Page for more information.

Note:
Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

URL Debugging

Allows you to enter a URL to debug an application on a server. Only server files will be debugged, so the files do not need to exist locally in your Workspace.

See Debugging a URL for more information.

Note:
Your server must be running the Zend Debugger in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

Toolbar Debugging

Allows you to Debug files and applications directly from your browser using the Zend Debugger Toolbar, available for download from the Zend Studio Resources site (http://www.zend.com/en/products/studio/downloads).

See Debugging Using the Zend Debugger Toolbar for more information.

Profiling

The Zend Profiler displays a breakdown of the executed PHP code in order to detect bottlenecks in scripts by locating problematic sections of code. These are scripts that consume excessive loading-time. The Profiler provides you with detailed reports that are essential to optimizing the overall performance of your application.

Zend Studio includes five different profiling methods:

PHP Script Local Profiling

Allows you to profile files on your workspace using Zend Studio's internal debugger.
The Internal Debugger enables developers to locally validate freshly developed code before deploying to a web server. The internal option means that only files located in local directories can be profiled. When profiling internal files the Zend Studio Internal Debugger uses its own PHP executable that was installed together with Zend Studio.

See Locally Profiling a PHP Script for more information.

PHP Script Remote Profiling

Allows you to profile files on your workspace using your server's Zend Debugger.
This allows you to profile local files using the Zend Debugger situated on your server. This allows you to test your files in the production environment, and allows you to utilize the extensions installed on your server.

See Remotely Profiling a PHP Script for more information.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

PHP Web Page Profiling

Allows you to profile applications situated on a server. It allows you to profile whole applications and projects.
The PHP Web Page Profile setting has an option to give the files you are working on first priority when profiling, using the ”Local Copy” option. This means that, when possible, file content is taken from the files situated on your Workspace. This prevents you from having to upload the latest revisions.

Note:
It's recommended that your local project structure reflect the project structure on your server.

See Profiling a PHP Web Page for more information.

Note:
Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.
The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

URL Profiling

Allows you to enter a URL to profile an application on a server. Only server files will be profiled, so the files do not need to exist locally in your Workspace.

See Profiling a URL for more information.

Once a Profile session has been executed (see Using the Profiler for more information on how to execute a profiling session), various views in the Profiling Perspective provide information on the performance of your script. See PHP Profile Perspective for more information on the various views.

Toolbar Profiling

Profile files and applications directly from your browser.

See Profiling Using the Zend Debugger Toolbar for more information.

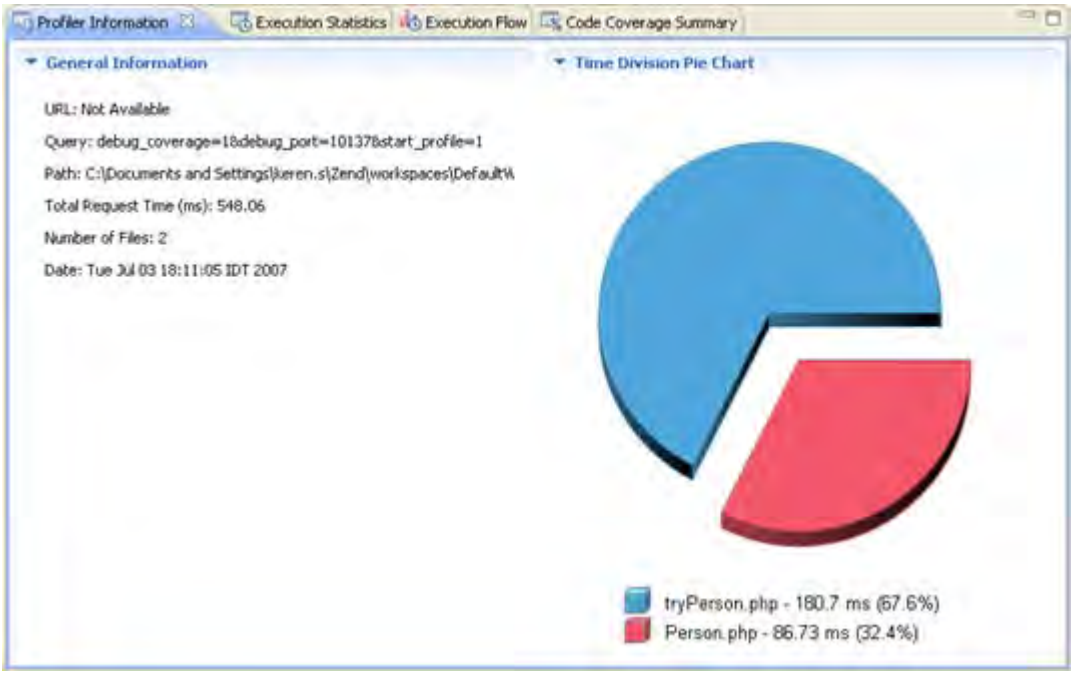
Profiler Views

Profiler Information view

Provides general information on the profiling duration and date, number of files constructing the requested URL and more. In addition, it displays a Time Division Pie Chart for the files in the URL.

The right side displays time division in a pie chart and the left side provides the following information:

- URL - The URL analyzed (if applicable)
- Query - The specific query parameters
- Path - The location of the first file called
- Total Request Time - Total processing time for the entire page
- Number of Files - Number of files processed
- Date - Date and time that the profiling took place



Execution Statistics view

Displays the list of files that were called during the profiling process and detailed information on processing times for elements within the files. The window contains statistics relevant to each element as follows:

- Function - The name and location of the function.
- Calls Count - The number of times that the function was called.
- Average Own Time - The average duration without internal calls.
- Own Time(s) - The net process duration without internal calls.
- Others Time(s) - Time spent on calling other files.
- Total Time(s) - The total time taken to process.

Function	Calls Count	Average Own Time	Own Time(s)	Others Time(s)	Total time(s)
tryPerson.php					0.180701
main	1	0.180701	0.180701	0.086728	0.267429
Person.php					0.086728
Person					0.086722
__construct	3	0.028747	0.086240	0.000116	0.086356
getId	3	0.000004	0.000013	0.000000	0.000013
setFirstName	3	0.000016	0.000047	0.000000	0.000047
getFirstName	3	0.000003	0.000009	0.000000	0.000009
setLastName	3	0.000005	0.000016	0.000000	0.000016
getLastName	3	0.000003	0.000010	0.000000	0.000010
setAge	3	0.000006	0.000017	0.000000	0.000017
getAge	3	0.000004	0.000013	0.000000	0.000013

Note:

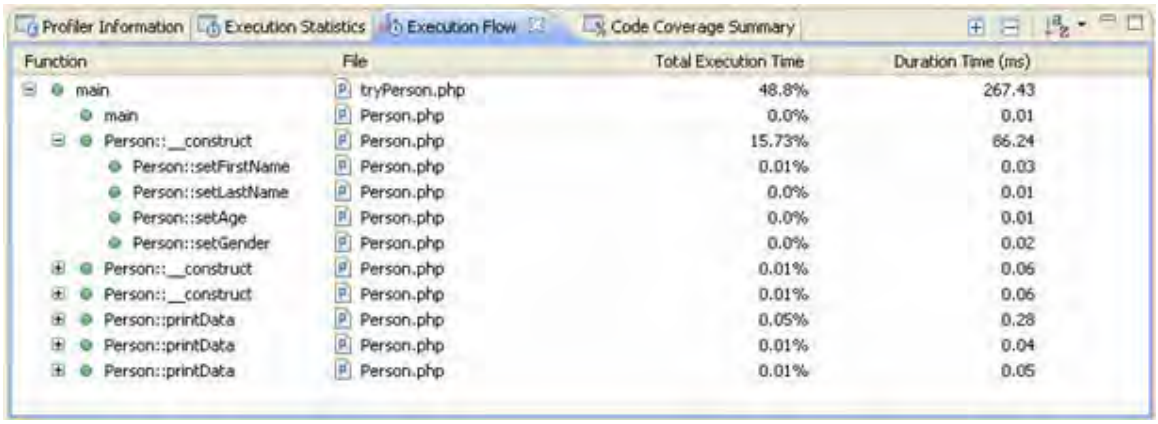
Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.

Right- clicking a function in the list gives you the option to 'Open Function Invocation statistics'. This will open a view with statistics about the selected function, the functions it was invoked by and functions that it invoked.

Execution Flow view

Shows the flow of the execution process and summarizes percentages and times spent on each function.

- Function - Function name
- File - The file in which the function is located
- Total Execution Time - Percent of time taken per function.
- Duration Time - Time taken per function in milliseconds.



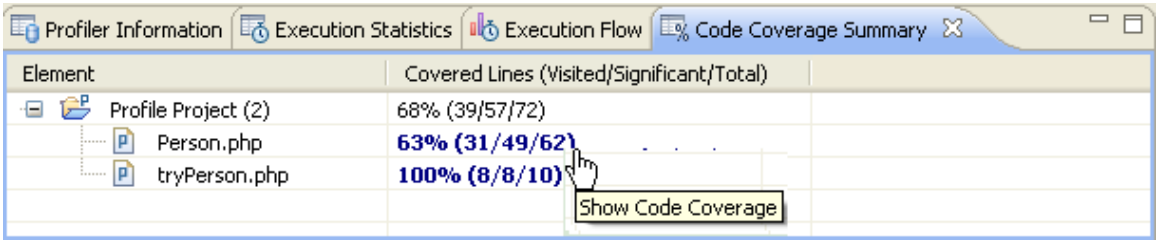
Right-clicking a function in the list gives you the option to:

- View Function Call - Will open the selected function call in the editor.
- View Function Declaration - Will open the selected function declaration in the editor.
- Open Function Invocation statistics - Will open a view with statistics about the selected function, the functions it was invoked by and functions it invoked.

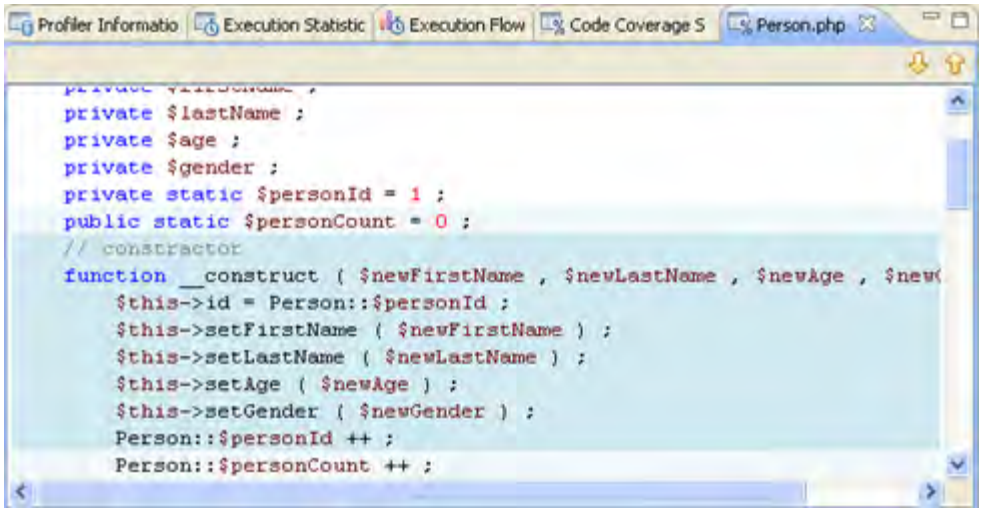
Code Coverage Summary

Summary of how many lines of code were covered during the Profiling process.

- Element - The file / project that was called.
- Covered Lines (Visited / Significant / Total) - Percentage of lines covered within each file. (Visited = Number of lines covered / Significant = number of lines that were significant to the function call / Total = Total number of lines in the file.)



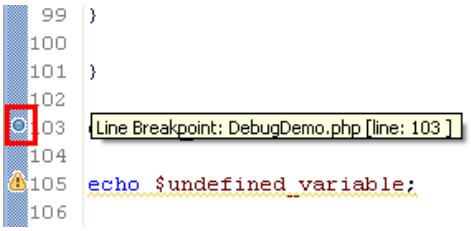
Clicking on the 'Covered lines' percentages will open an editor containing the file, with the covered lines highlighted:



Breakpoints

Breakpoints allow you to set places in the code at which the debugging process will pause. Setting a breakpoint in your script will cause the debugger to stop at the specified line, allowing you to inspect it.

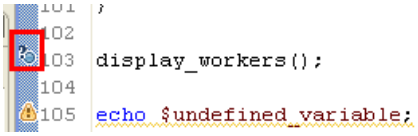
Breakpoints are represented by a blue ball in the vertical ruler to the left of the editor.



During the debugging process, breakpoints can be monitored in the Breakpoints View.

Conditional Breakpoints

Conditions can be added to breakpoints so that the debugger will only stop at them under certain conditions. These can be any valid boolean expressions.
Conditions can be added to breakpoints through the PHP Conditional Breakpoints Properties dialog. Conditional Breakpoints are represented by a blue ball with a question mark.



Example:

If a breakpoint has a condition of "$var > 5", the debugging process will stop at the breakpoint only if the value of $var is greater than 5.

PHP Include Paths

The PHP Include Path is a set of locations that is used for finding resources referenced by include/require statements.

Elements added to a project's Include Path affect the following:

- Running/Debugging/Profiling - Files that are called with a 'relative path' will be searched for during runtime in the resources and order specified in the include path.
- Go to source / content assist for include statements - Files that are called with a 'relative path' will be searched for according to the resources and order specified in the project's include path.
- Content Assist - Adding projects/libraries to a project's Include Path will make elements defined within the included projects/libraries available as Content Assist options to the project.

In 'include'/'require' calls, file locations can be defined in three ways:

a) **Absolute Path**- The exact file location is specified (e.g. C:\Documents and Settings\MyProject\myfolder\a.php). During Remote PHP Script (PHP Server) or PHP Web Page Running/Debugging/Profiling when the 'Local Copy' option is selected under the 'Source Location' category in the Advanced tab, the Path Mapping mechanism will be activated.

b) **Relative to the Current Working Directory** - File names preceded with a "./" or a "../" These will only be searched for relative to the PHP 'Current Working Directory'. You can find out the location of your Current Working Directory by running the command "echo getcwd()".

c) **Relative Path** - Only the file name or partial path is specified (e.g. /myfolder/a.php). In this case, Zend Studio will search for the file's path according to the resources and order configured in the project's Include Path.

   If the path of the file being searched for exists in more than one location, the file that is called will be the first one Zend Studio encounters during the search process.

   By default, the order in which Zend Studio searches for the file's path is as follows:

   i. Projects in the "debug target" (the first file to be debugged) project's Include Path, according to the order in which they are listed. If a project specified in the Include Path refers to other projects/libraries in its own Include Path, the file path will be searched for there before the search process continues.

   ii. The "debug target" file's project.

See http://il2.php.net/manual/en/function.include.php for more on PHP's search mechanism.

PHP Build Path

The PHP build process scans all resources that are on the project's PHP Build Path so that elements defined within them can be made available for Content Assist options and Refactoring operations. This is done in order to get notification about changes in the file system (e.g. files added/removed from the project, code changes etc.) and in order to maintain the code database (user classes, functions, variables etc.).

Configuring the project's Build Path allows you to select PHP resources to include/exclude from this process. Rather than automatically scanning all resources within the project, configuring the Build Path allows you to select which resources will be scanned (letting you, for example, exclude folders containing images, JavaScript files or other types of files not containing PHP code). This can significantly speed up the build process.

Path Mapping

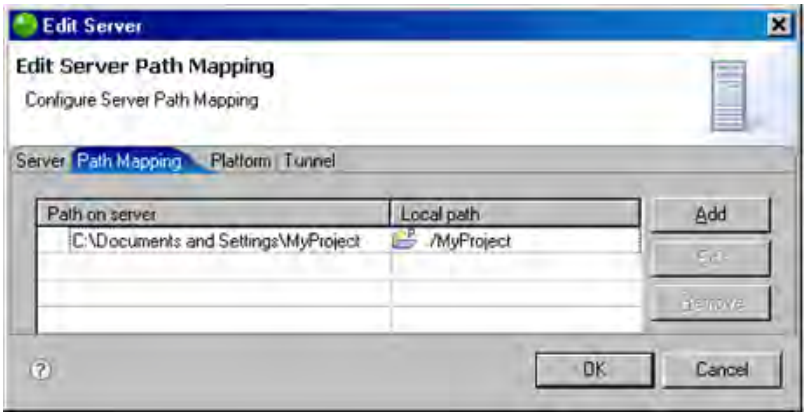
Zend Studio enables you to map server paths to local paths while Debugging and Profiling on a server. Once a Path Map has been defined, if a file is called from the defined location on the server during debugging/profiling, its content will be taken from the set corresponding location on the file system/workspace.

Note:

Path Mapping is only activated during Remote PHP Script (PHP Server) Debugging/Profiling, or PHP Web Page Debugging/Profiling when the 'Local Copy' option is selected under the 'Source Location' category in the Advanced tab.

Example:

The server path 'C:\Documents and Settings\MyProject' has been mapped to '/MyProject' on the Workspace:



During Remote PHP Script Debugging, a file is called from location 'C:\Documents and Settings\MyProject\a.php':

```
<?php
echo "a.php";
include ('C:\Documents and Settings\MyProject\a.php');
?>
```

The file content for a.php will be taken from the a.php file located in the 'MyProject' project, situated on the Workspace.

Note:

Server Path Maps can be viewed and defined in the Path Mapping tab of the PHP Servers Preferences page.

Defining Path Maps

Path Maps can be defined in three ways:

1. Manually, through the PHP Servers Preferences page. See 'Adding a Server Location Path Map' for more information.
2. Automatically whenever a file is debugged/profiled - A Path Map is automatically set between the path to the debug target's parent project (the parent project of the file from which the debugging process has been launched - e.g. C:\Workspace\MyProject) and the debug target's project in the Workspace (e.g. MyProject).
3. Through the Path Mapping dialog. This is launched during debugging/profiling whenever a file defined with an absolute path (See 'Include Paths' for more on absolute file locations) is called.

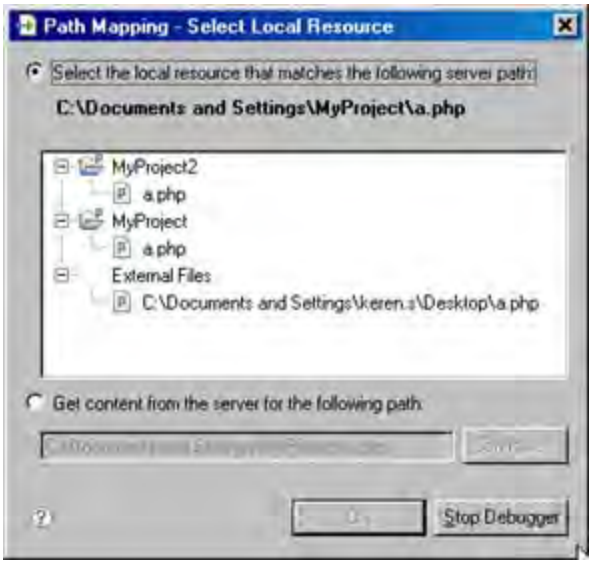
   In this scenario, a Path Mapping dialog will appear with a list of 'similar files' to the one being called.

   'Similar' files are files with the same name as the called file that are situated in the following locations:

   - Files in the project from which the file was called.
   - Files in projects that are in the Include Paths of the project from which the file was called.
   - Files that are open in a Zend Studio editor.

Note:

If the debug/profile session was triggered from the Zend Debugger Toolbar, all files in the Workspace with the same name will be listed.



Note:

The dialog will not appear if a Path Mapping to the called location has already been defined.

Selecting a file from the list results in a Path Map being created between the called remote file's parent folder and the parent folder of the 'similar' file selected from the list. This means that every time a file is called from the same parent folder, it's content will be taken from the file situated in the selected Workspace/local folder.
If none of the options in the Matching items list represent your desired file location, you may select the 'Get content from the server for the following path' option. This means that whenever this path is called during the debugging/profiling process, it will only be searched for on the server. (This is done using PHP's search mechanism - see http://il2.php.net/manual/en/function.include.php for more information..)
You can click Configure to modify the path to include any parent or child directories.

Note:
Selecting the 'Get content from the server for the following path' option will only affect the current Debug / Profile session. No Path Mapping will be defined.

Zend Browser Toolbar

The Zend Browser Toolbar is an external add-on to Internet Explorer and Firefox browsers which allows you to initiate debugging or profiling sessions directly from your browser. Debugging / profiling through the Zend Browser Toolbar will launch a remote debug / profile session in Zend Studio.

The Zend Browser Toolbar contains the following options:

- **Zend Studio** - Opens Zend Studio. Ensure the Zend Studio .exe file is configured in the Zend Toolbar Settings.
- **Debug** - Launches a Debugging session in Zend Studio.
- **Profile** - Launches a Profiling session in Zend Studio.
- **Extra Stuff** - Provides access to Zend Browser Toolbar settings, and links to useful Zend and PHP Information.
- **Search for PHP Information** - Allows you to quickly and easily search the web for PHP information.

See Installing and Configuring the Zend Browser Toolbar for information on how to get started with the Zend Browser Toolbar.

Tunneling

Tunneling provides a means of persistent connection between Zend Studio and a remote server situated behind a Firewall or NAT. After creating a Tunnel, all communication between Zend Studio and the server can go through that tunnel, instead of assigning more communication ports for the Debug/Profile Sessions made on the remote server.

Note:

A communication tunnel cannot currently be created to a Windows server.

[This persistent connection operates even when separated by a firewall. The advantage of this method is that it is possible to use the Zend Studio Integration on several nodes at once. For example, debugging an entire cluster of machines behind a load-balancer over a single debugger connection to Zend Platform's Studio Server component.

The technology is based on two functional elements:

- The Zend Studio that includes an internal Web server that listens on the local host on a specific Auto Detection port.
- Zend Platform auto-evaluates Zend Studio's Auto Detection port, by evaluating Zend Studio's settings. These are the Tunnel Settings that are defined in Zend Platform]

The tunnel communication port should be used in the following circumstances:

1. When debugging or profiling files on a remote server which is behind a firewall or other security device.

Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

2. Establishing communication between Zend Studio and Zend Server when Zend Server is running on a remote server which is behind a firewall or other security device. The communication between Zend Studio and Zend Platform facilitates the integration that combines Zend Server's event reporting capabilities with Zend Studio's editing, debugging and profiling features. This enables the viewing and debugging/profiling of Zend Server events in Zend Studio.

   See Zend Server Integration for more information.

Note:

To find out whether your server is situated behind a firewall, contact your System Administrator.

To set up a tunneling connection, several configuration settings need to be defined both in Zend Studio and on your server's debugger (this can be done through Zend Server or your php.ini file). See Setting Up Tunneling for more information.

Zend Server

Zend Server is a complete, enterprise-ready Web Application Server for running and managing PHP applications that require a high level of reliability, performance and security. It includes the most reliable and up-to-date version of PHP, tested PHP extensions, database drivers and other enhancements. Zend Server comes bundled with Zend Framework (the leading open-source PHP framework), Apache and MySQL.

Zend Server provides an optimal environment for developing and deploying your PHP applications.
See Setting Up Remote Debugging for information on how to set up your Zend Server for debugging with Zend Studio.

See the Zend Server site (http://www.zend.com/products/server/) to learn more about and download Zend Server.

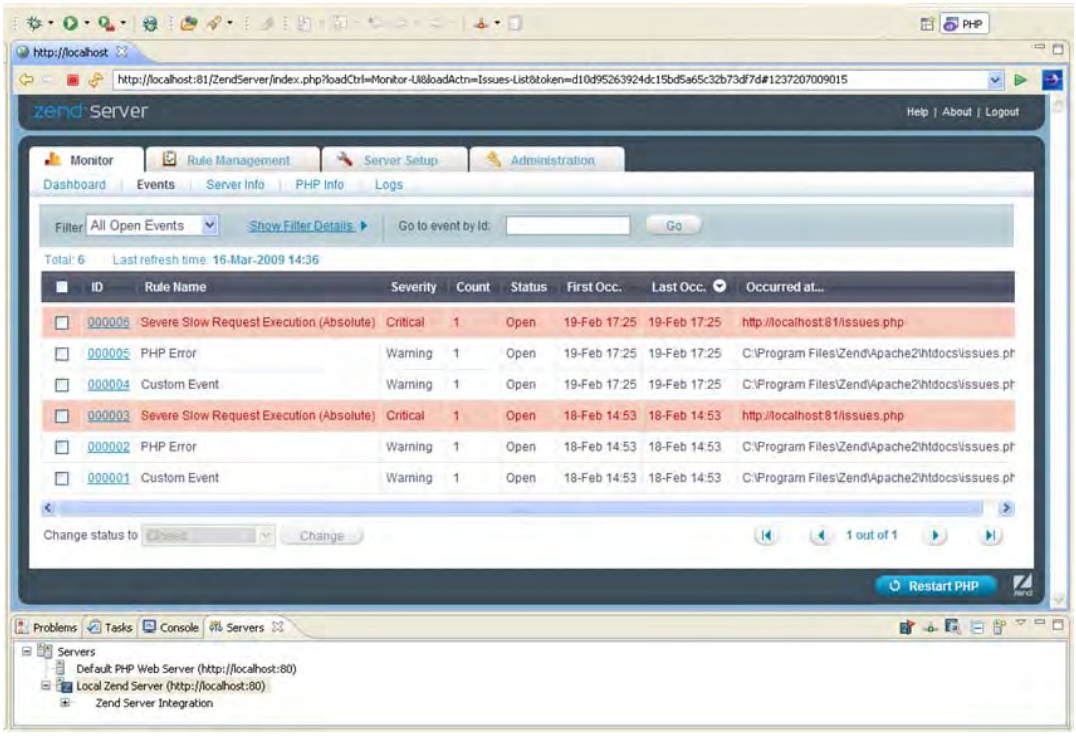
Zend Server Integration

Integrating Zend Studio with Zend Server allows you to benefit both from Zend Studio's debugging and profiling functionality and from Zend Server's Event Monitoring capabilities. Zend Server monitors and constantly tests your PHP environment and programs in order to allow you to gain maximum efficiency. Instances of problematic scripts and slow execution are captured by Zend Server as 'events.' Zend Server's integration with Zend Studio means that the problems identified by Zend Server can then be viewed, tested, debugged and profiled in Zend Studio. This can be done through the Zend Server user interface or by importing Zend Server Events into Zend Studio.

Viewing Events from Inside Zend Studio

Zend Server's events can be viewed and debugged directly from within Zend Studio using the internal browser, which displays the Zend Server User Interface.

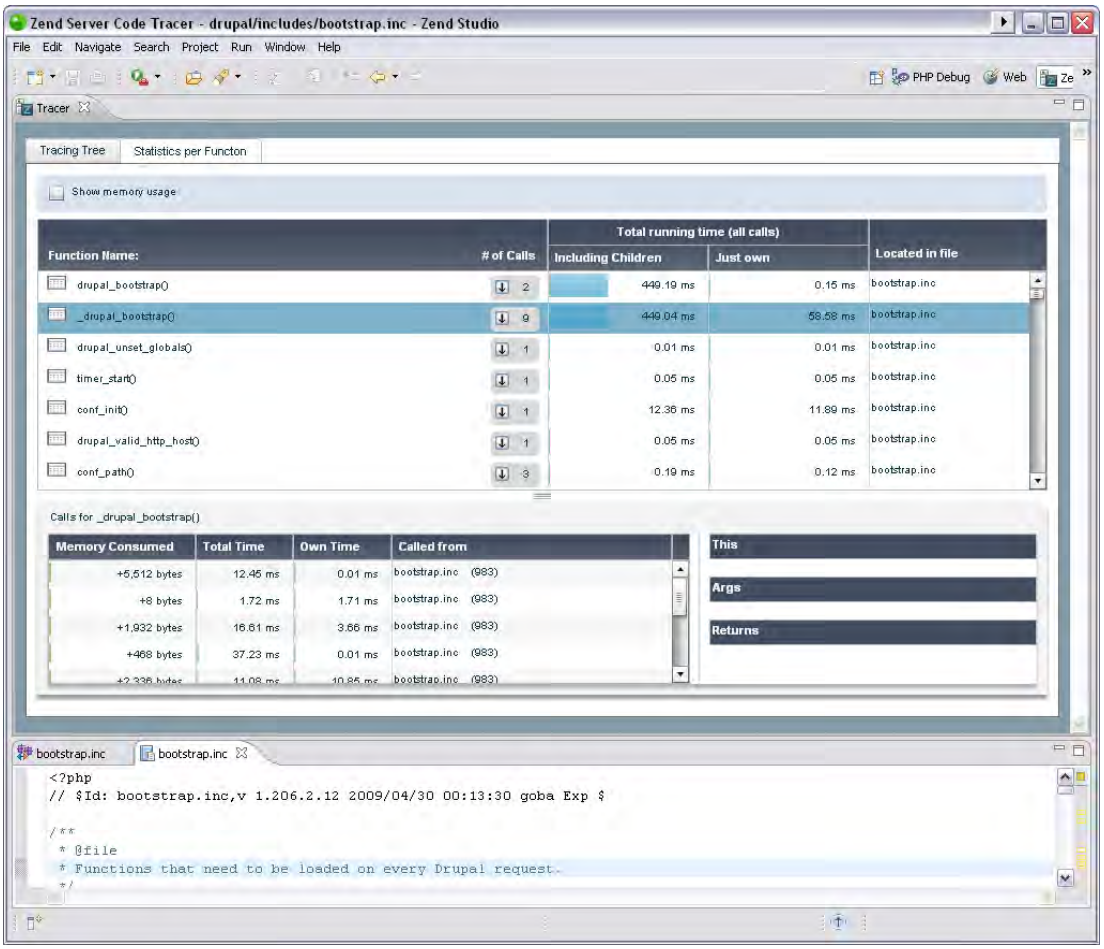
See Debugging / Profiling Events from Zend Server for more information.



Importing Events into Zend Studio

Zend Server events can be exported from Zend Server as a xml file, which can then be imported into Zend Studio and debugged. This is useful when the developer using Zend Studio does not have access to the Zend Server on which the event occurred.
See Importing Events Into Zend Studio for more information.

Zend Server events can also be imported as .amf files using the Code Tracing feature in Zend Studio. Integrating Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. Code Tracing uses the Zend Server Code Tracer Perspective to allow you to view the trace data within your environment.
This feature is useful in resolving time performance issues, memory performance issues, and workflow errors.



Code Tracing

Zend Server Code Tracing captures full execution data (trace data) of PHP applications in real time. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure. Integrating Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. Code Tracing uses the Zend Server Code Tracer Perspective to allow you to view the trace data within your environment.

This feature is useful in resolving time performance issues, memory performance issues, and workflow errors.

In order to use the Code Tracing feature, you must first export a Zend Server Event File from Zend Server and then import the Zend Server Event File into Zend Studio. Once the Zend Server Event File has been imported, you can open the source of trace data.

Important Note:

For more information about additional features and usability of Code Tracing see Code Tracing in the Zend Server Online Documentation.

The Code Tracing feature in Zend Studio allows you to do the following:

- Export Trace Information (Located in the Zend Server Online Documentation)
- Import a Zend Server Event File
- Open the Source of Trace Data

For more information see Working with Code Tracing.

Note:

The source of the execution data can only be opened in an environment in which the project already exists locally.

PHPUnit Testing

Unit testing is a procedure to test your code to ensure that individual units of source code are working properly and that the right output is being generated. Tests can be run on all or some functions within files, meaning that tests can be conducted before the file has been fully developed. Each test case should be independent of others to ensure that test results can pinpoint the location of the error.

Running unit tests can ensure that your code is stable and functioning correctly, and can help you to diagnose errors.

PHPUnit Test Cases

PHPUnit Test Cases can be created for each class within a file. Running a PHPUnit Test allows you to see which functions within the test are working correctly.
See Creating a PHPUnit Test Case and Running a PHPUnit Test Case for more information.

PHPUnit Test Suites

PHPUnit Test Suites can be created to run several PHPUnit test cases at once. See Creating a PHPUnit Test Suite and Running a PHPUnit Test Suite for more information.

Running PHPUnit Test Cases/Suites

Running a PHP Unit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run.

Debugging PHPUnit Test Cases/Suites

Debugging a PHPUnit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run, in addition to the normal debug functionality. This will also allow you to debug and analyze PHPUnit libraries.

Profiling PHPUnit Test Cases/Suites

Profiling a PHPUnit Test Case/Suite will result in the PHPUnit view being displayed, showing the results of the tests that were run, in addition to the normal profiling functionality. This will also allow you to profile and analyze PHPUnit libraries.

PHPUnit Reporting

Once a PHPUnit Test has been run, Zend Studio can generate a range of reports to easily view and analyze your tests.

The types of reports available are:

- XML - Generates an XML output of your test results. This can be used to create your own reports.
- 'plain.xsl' - Creates a report based on a predefined 'plain.xsl' format. The report shows It shows percentages, elapsed time, total tests, errors and failures. In addition it shows individual report status, message, stack trace and warnings with their stack traces.
- 'packages.xsl' - Creates a report based on a predefined 'packages.xsl' format. This is an extension of the plain report which divides the unit tests into 'packages'. Packages are defined by adding an "@package" annotation to the PHPDoc of test classes. Test classes without an @package annotation will be categorized in a 'default package'.

Custom XSL reports can also be generated by selecting the 'Generate with XSL...' option and selecting a previously created XSL report. These will then be added to the list of available reports.

See Reporting on PHPUnit Test Results for more information.

Refactoring

The Refactoring feature in Zend Studio allows you to:

- Rename and move files and elements within those files, while maintaining the links between the items. Once an element or file has been renamed or moved, all instances of that item within the project will be automatically updated to reflect its new name / location.
- Extract variables - Creates new variables for expressions.
  Creates a new variable assigned to the expression currently selected and replaces the selection with a reference to the new variable.

Example:

Implementing an extract method refactoring on the variable $my_form in the following code:

```
<?php
if ($this->getRequest()->isPost()) {

            if ($my_form->isValid($request->getPost())) {

                $model = new Default_Model_Guestbook($my_form->getValues());
                $model->save();
                return $this->_helper->redirector('index');
            }
        }
        if (!$my_form->isValid($request->getPost())) {

            $model = new Default_Model_Guestbook($my_form->getValues());
            $model->load();
        }
```

Will result in the following code being created (the changes have been highlighted):

```
<?php
$isValid = $my_form->isValid($request->getPost());
if ($this->getRequest()->isPost()) {

            if ($isValid) {

                $model = new Default_Model_Guestbook($my_form->getValues());
                $model->save();
                return $this->_helper->redirector('index');
            }
        }
        if (!$isValid) {

            $model = new Default_Model_Guestbook($my_form->getValues());
            $model->load();
        }
```

- Extract Methods - Creates a new method to replace all occurences of a selected fragment of code.

Martin Fowler, the creator of the Refactoring concept, defines it as the following:
"Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior. Its heart is a series of small behavior preserving transformations. Each transformation (called a 'refactoring') does little, but a sequence of transformations can produce a significant restructuring. Since each refactoring is small, it's less likely to go wrong. The system is also kept fully working after each small refactoring, reducing the chances that a system can get seriously broken during the restructuring."
http://www.refactoring.com

JavaScript Support

JavaScript is a scripting language designed to add interactivity to HTML pages.
Zend Studio provides support for JavaScript features in standalone JavaScript files as well as in PHP files and projects.

Some of the JavaScript features available for PHP projects and files are:

- JavaScript Element Outline - JavaScript elements will be displayed in the Outline view.
- JavaScript libraries will be added to the Include Paths of PHP projects for which JavaScript support has been enabled.
- JavaScript Build Paths - Setting the JavaScript Build Path for a project will allow you to determine which elements will be included in the project's build process. This allows access to content assist and refactoring options from resources in locations other than in the project itself.
- Advanced editing functionality in JavaScript editors. For example:
  - JavaScript Content Assist
  - JavaScript Syntax Coloring
  - JavaScript Code Commenting
  - JavaScript Hover Support
  - JavaScript Mark Occurrences
  - JavaScript Smart Goto Source

JavaScript support can be added to new or existing PHP projects so that all of Zend Studio's JavaScript features are available to the project.
For a full list of features enabled for JavaScript development, see Features - Javascript Developtment Toolkit (JSDT) in the JavaScript Development Toolkit User Guide.

Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

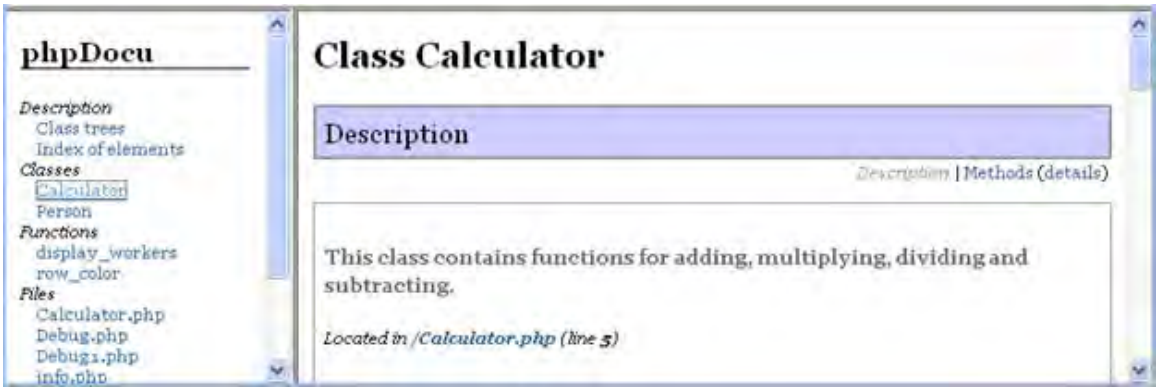
PHPDocs

PHPDocs provides structured, easy-to-read documentation of all your php elements.

PhpDocumentor can automatically create PHPDocs from your scripts, using a templating system to change your source code comments into readable formats.
The PHPDoc Generator Wizard is Zend Studio's interface with phpDocumentor.

PHPDocs list all classes, functions, files and other elements in an easily-browsable format so that scripts can be easily navigated and understood.

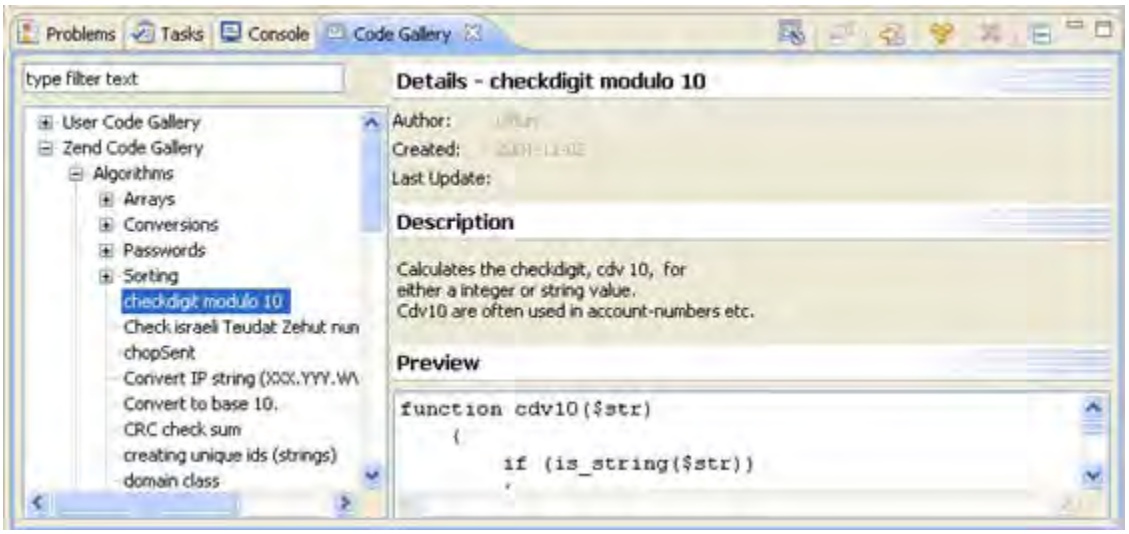
PHPDocs also incorporate PHPDoc Block comments to provide descriptions and parameters for your code elements:



Code Galleries

Code snippets can be used to easily insert pre-defined sections of code into your script.
Code Galleries are pre-defined code snippets sites. Connecting to a Code Gallery will give you access to a selection of code snippets.

The Code Gallery view allows access to the Code Gallery sites and code snippets contained within them so that the code snippets can be easily inserted into your script.



The Code Gallery view comes by default with a User Code Gallery, to which your own code snippets can be added, and the Zend Code Gallery. Access to the Zend Code Gallery will require registration to the Zend Network.

Selecting a code snippet will result in its details, description and preview being displayed in the left of the view.

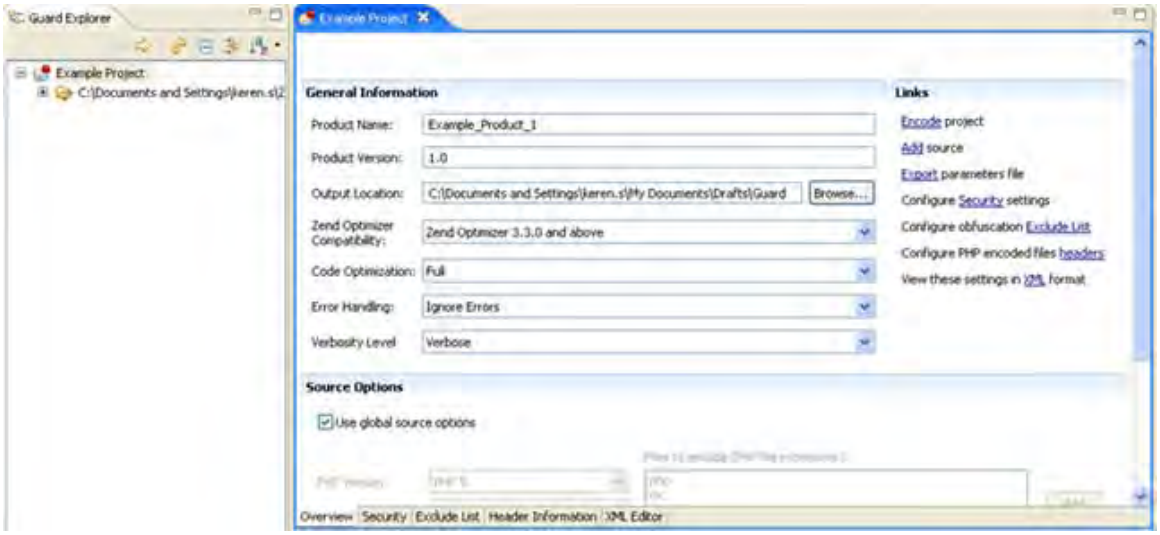
Zend Guard Integration

Zend Guard is the first Electronic Licensing solution for the PHP marketplace. It includes the Encoding solution that pioneered PHP intellectual property protection. Unprotected intellectual property, in the form of plain text PHP scripts and software without license restrictions, can be copied, modified, and retained by someone else. It is available to your competitor, to hackers and even to developers at customer sites.

Zend Guard provides tools that significantly lessen risk to your intellectual property. It is designed to prevent your property from being viewed or modified.

See the Zend Guard product site (http://www.zend.com/en/products/guard) or the Zend Guard Online Documentation (http://files.zend.com/help/Zend-Guard/zend-guard.htm) for more information on Zend Guard.

Zend Studio's integration with Zend Guard allows you to apply Zend Guard's encoding functionality to projects and applications created and stored in Zend Studio by allowing you to open them in Zend Guard. Conversely, files stored in Zend Guard can be opened and edited in Zend Studio.



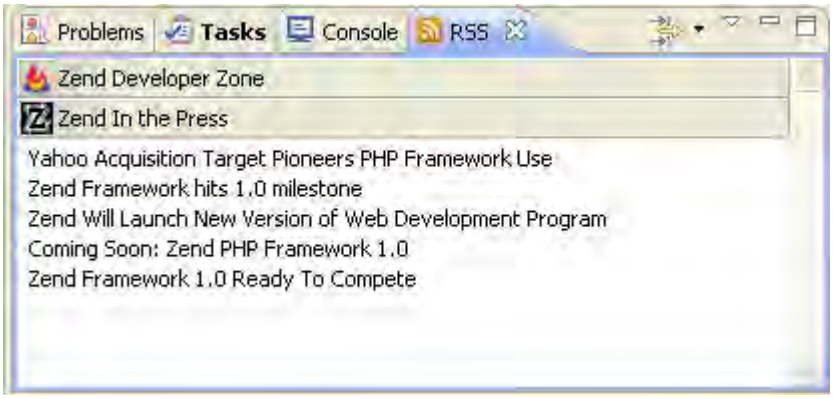
RSS Feeds

RSS is a format for syndicating news feeds from news sites and weblogs.

Zend Studio has a built-in RSS reader allowing you to view news and updates from the Zend Developer Zone, to view the latest mentions of Zend in the press or to view any other RSS feed channel.

RSS feeds are displayed as a list in the RSS view and can be displayed in Zend Studio's internal browser.

Each new item will be listed according to the RSS channel or time.


Problems
Tasks
Console
RSS
Zend Developer Zone
Zend In the Press
Yahoo Acquisition Target Pioneers PHP Framework Use
Zend Framework hits 1.0 milestone
Zend Will Launch New Version of Web Development Program
Coming Soon: Zend PHP Framework 1.0
Zend Framework 1.0 Ready To Compete

WSDL - Web Services Description Language

WSDL (Web Services Description Language) is an XML-formatted language used to describe Web service capabilities. Web services are a standardized way of allowing applications to interface and share data across the network. Web service messages are written in XML, thus allowing for different applications in different programming languages to interface with each other.

WSDL files define how the Web services work and the operations they perform. Zend Studio provides an integrated means for incorporating and inspecting WSDL files and a wizard for generating your own WSDL files.

Zend Studio for IBM i Extras

Note:

The features listed below are only available in the Zend Studio for IBM i.

Zend Studio for IBM i contains additional extras to help you connect to and use IBM i functionality.

Zend Studio contains the following extras:

1. The ability to register for a free Zend Studio for IBM i Edition license
2. Templates for IBM i PHP API Toolkit functions
3. Content Assist for:
   - The Zend 5250 Bridge
   - IBM i PHP API Toolkit functions

Free Registration

System i users who have downloaded Zend Studio are entitled to a free license.

To receive and activate your free license:

1. To go to the Zend registration website go to the Menu Bar | Help | Get a License.
2. In the site, enter your System i server serial number and optionally check the box to have the license emailed to you.
3. Click the Generate License button.
4. When prompted, enter you Zend User ID and Password.
   The license key will be generated and you will see your User Name and License Key displayed.
5. In Zend Studio, from the Menu bar go to Help | Register and enter your User Name and License Key.
6. Click OK.

Your Zend Studio for IBM i will be registered and all functionality will be available.

IBM i PHP API Toolkit functions Templates

For information on Templates and how to use them, see the 'Using Templates' topic.

Zend Studio contains templates for the following IBM i PHP API Toolkit functions:

IBM i Template	Explanation
i5ActiveJobs	Enables retrieving the system's active jobs: 1. Connects to i5 server 2. Opens active job list 3. Gets array for an active job entry 4. Closes handle received from i5_job_list function 5. Closes connection to i5 server
i5Connect	Enables connecting to the i5 server: 1. Connects to i5 server 2. Closes connection to i5 server
i5DataAreaCreate	Creates the data area: 1. Connects to i5 server 2. Creates data area of given size 3. Closes connection to i5 server
i5DataAreaDelete	Enables deleting the data area: 1. Connects to i5 server 2. Deletes data area 3. Closes connection to i5 server
i5DataAreaRead	Enables reading from a data area: 1. Connects to i5 server 2. Reads from data area 3. Closes connection to i5 server
i5DataAreaWrite	Enables reading from a data area: 1. Connects to i5 server 2. Reads from the data area 3. Closes connection to i5 server
i5DtaqReceive	Enables reading data from the data queue without key: 1. Connects to i5 server 2. Reads data from the data queue without key 3. Closes connection to i5 server
i5DtaqReceiveKey	Enables reading data from the data queue with key: 1. Connects to i5 server 2. Reads data from the data queue with key

IBM i Template	Explanation
	3. Closes connection to i5 server
i5DtaqSend	Enables putting data to the data queue without key: 1. Connects to i5 server 2. Puts data to the data queue without key 3. Closes connection to i5 server
i5DtaqSendKey	Enables putting data into the data queue without a key, it 1. Connects to i5 server 2. Puts data to the data queue without key 3. Closes connection to i5 server
i5JobLogs	Enables retrieving job log entries: 1. Connects to i5 server 2. Opens job log 3. Gets array for a job log entry 4. Closes handle received from i5_jobLog_list function 5. Closes connection to i5 server
i5ObjectListing	Enables getting an array with the message element for an object list entry: 1. Connects to i5 server 2. Opens object list 3. Gets for a object list entry 4. Closes handle received from i5_objects_list function 5. Closes connection to i5 server
i5Program	Enables calling a program and accept results from it: 1. Connects to i5 server 2. Opens a program or service procedure and prepares it to be run 3. Calls the program and optionally accepts results 4. Free program resource handle 5. Closes connection to i5 server
i5ProgramService	Creates Web Services class enabling invoking an RPG program: 1. Connects to i5 server 2. Opens a program or service procedure and prepares it to be run 3. Calls the program and optionally accepts results

IBM i Template	Explanation
	4. Free program resource handle 5. Closes connection to i5 server
i5Spool	Enables getting spool file data from the queue and getting the data from the spool file: 1. Connects to i5 server 2. Creates an pool file lists, of certain output queue or for all queues 3. Gets spool file data from the queue 4. Get the data from the spool file 5. Free spool list resource 6. Closes connection to i5 server
i5UserSpaceCreate	Creates a new user space object: 1. Connects to i5 server 2. Creates new user space object 3. Closes connection to i5 server
i5UserSpaceDelete	Enables deleting a user space object: 1. Connects to i5 server 2. Deletes user space object 3. Closes connection to i5 server
i5UserSpaceGet	Retrieves user space data: 1. Connects to i5 server 2. Opens a user space and prepares it to be run 3. Retrieves user space data 4. Closes connection to i5 server
i5UserSpacePut	Enables to add user space data: 1. Connects to i5 server 2. Opens a user space and prepares it to be run 3. Adds user space data 4. Closes connection to i5 server

Content Assist

Zend Studio, contains Content Assist for commonly used IBM i functionality. Content Assist is available for the IBM i PHP Toolkit functions (listed above), as well as for connectivity to the Zend 5250 Bridge.

For more information on Content Assist and how to use it, see the Content Assist topic.

Remote System Explorer

RSE (Remote System Explorer) provides transparent access to remote resources, including the upload and download of files. RSE helps you to create, view and manage your connections to remote systems such as FTP and SSH. This will allow you to work on projects with local files remotely enabling you to create, view and manage your connections to remote systems.

RSE enables you to perform the following tasks:

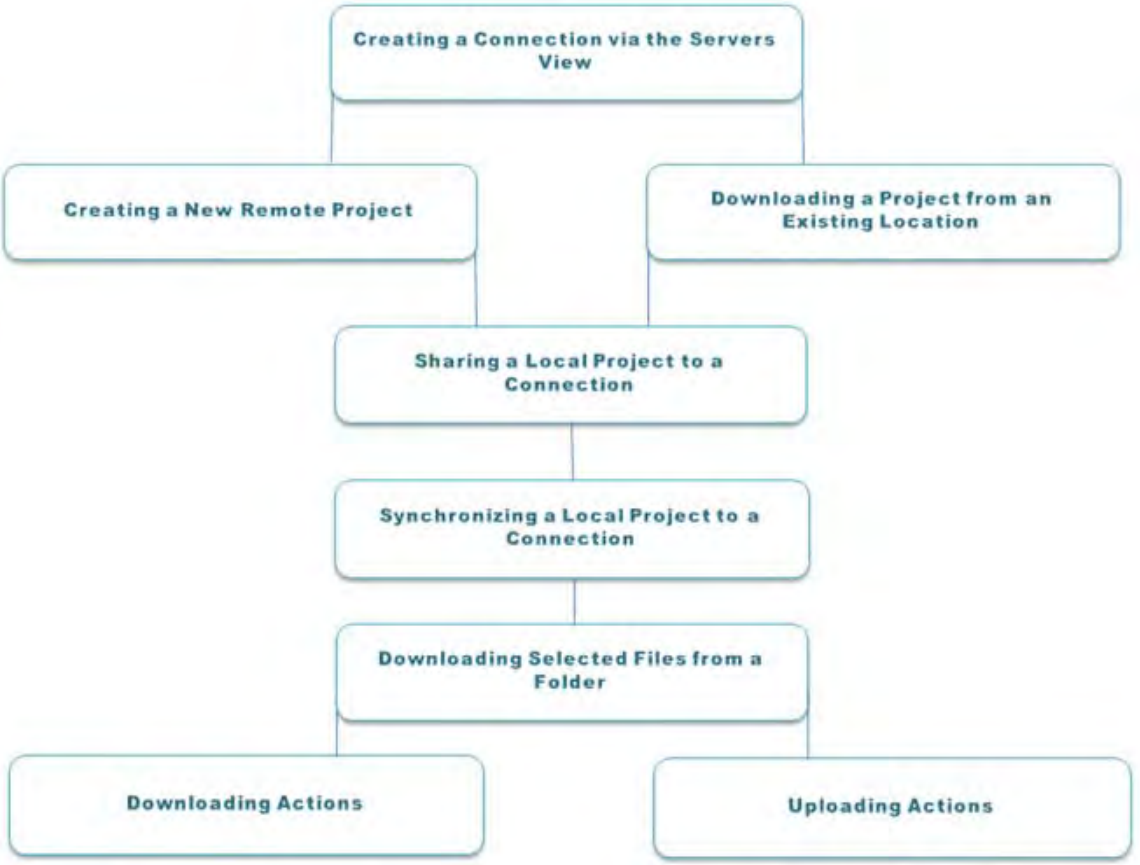
- Create a connection via the Server's view.
- Create a new remote project.
- Import a project from an existing connection.
- Share a local project to a connection.
- Synchronize a local project to a connection
- Export Actions
- Import Actions

RSE is an upgraded version of the Remote Systems View that reduces the amount of communication between networks, by using less connection nodes. RSE also communicates less data, by directly transferring information rather than copying all of the files.

Remote Server Support

RSS (Remote Server Support) provides transparent access to remote resources, including the upload and download of files. RSS helps you to create, view and manage your connections to remote systems such as FTP and SSH. This will allow you to work on projects with local files remotely enabling you to create, view and manage your connections to remote systems.
RSS enables you to perform the following tasks:

- Creating a Connection via the Server View - Create different types of connections, for creating and managing projects.
- Creating a New Remote Project - This procedure describes how to create a new remote project. Once the remote server has been set up, you can create a new project which will be synchronized with one of the remote servers.
- Downloading a Project from an Existing Connection - This procedure describes how to download a project from the server from a connection that already exists.
- Sharing a Local Project to a Connection - This procedure describes how to share a project from your local server to a remote connection.
- Synchronizing a Local Project to a Connection - This procedure describes how to synchronize, download a local project from server and upload a local project to a server connection.
- Downloading Selected Files from a Folder - This procedure describes how you can import a partial project, ignoring any files that you don’t need and keeping them hidden on your workspace.


Creating a Connection via the Servers View
Creating a New Remote Project
Downloading a Project from an Existing Location
Sharing a Local Project to a Connection
Synchronizing a Local Project to a Connection
Downloading Selected Files from a Folder
Downloading Actions
Uploading Actions

RSS is an upgraded version of the Remote Systems View that reduces the amount of communication between networks, by using less connection nodes. RSS also communicates less data, by directly transferring information rather than copying all of the files.

Note:

The Remote Systems View page is now obsolete. Please use the following link to access the upgraded feature, Remote System Explorer Working Remote Server Explorer.

Mylyn Integration

Mylyn is a task-focused interface that reduces the overload of information and makes multi-tasking easy. It does this by making tasks a top priority, and integrating rich and offline editing for repositories such as Bugzilla, Trac, and JIRA. Once your tasks are integrated, Mylyn monitors your work activity to identify relevant information, and uses this task context to focus the user interface on the task-at-hand. This puts the information you need at your fingertips and improves productivity by reducing searching, scrolling, and navigation. By making task context explicit Mylyn also facilitates multitasking, planning, reusing past efforts, and sharing expertise.

The Task List allows you to view and manage your tasks. The Task List contains both "Local Tasks" and shared "Repository Tasks" that are stored in a task repository such as Bugzilla or Jira. Local tasks are typically contained in categories. Repository tasks are contained in special categories that represent queries.

Mylyn features the following:

- Task List Presentation
- Icon Legend and Color Coding
- Creating New Tasks
- Using Local Tasks
- Repository Tasks
- Task Focused Interface
- Task Focused Ordering
- Task Hyperlinking

For more information see the Mylyn User Guide.

Phar Integration

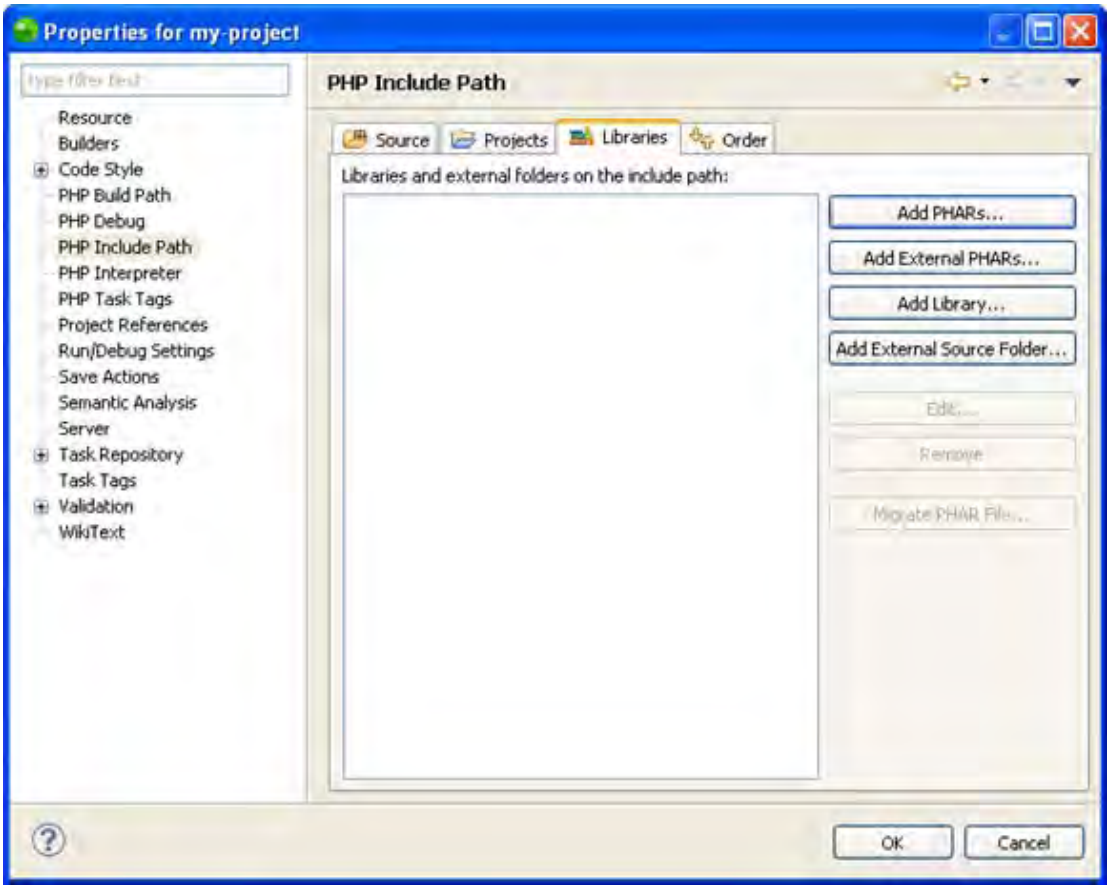
Phar Integration will enable you to integrate with phar archives. Phar is an archive system that enables you to group numerous files into a single file for easy distribution and installation. A phar archive provides a way to distribute a complete PHP application in a single file and run it from that file without needing to even use a disk. Using a Phar archive library is identical to using any other PHP library.

Phar Integration contains the following features:

- Export to phar.
- Import to phar.
- Add phar to your project.

Please click here for more information on Phar (external link).

Phar integration settings are defined per project and are part of the project library settings.



Tasks

Tasks

Creating PHP Projects	Connecting to Databases
Creating PHP Files	Running Files and Applications
Creating PHP Elements	Debugging Files and Applications
Migrating From Zend Studio 5.X	Profiling Files and Applications
Zend Server Integration	Managing PHP Libraries
Working with Code Tracing	Configuring a Project's PHP Include Path
Using Content Assist	Configuring a Project's PHP Build Path
Using Templates	Adding a Server Location Path Map
Formatting Code	Debugging and Profiling Zend Server Events
Using Code Folding	Using PHPUnit Testing
Searching for PHP Elements	Using Refactoring
Opening PHP Elements	Generating Getters and Setters
Opening Types/Methods	Overriding / Implementing Methods
Using Smart Goto Source	Creating a PHPDoc
Viewing Type Hierarchies	Creating and Opening HTML Files
Creating PHP Working Sets	Using Code Galleries
Using Mark Occurrences	Integrating with Zend Guard
Finding and Replacing	Working with WSDL
Applying Quick Fixes	Incorporating WSDL Files
Adding Comments	Viewing RSS Feeds
Adding PHP DocBlock Comments	Working with Remote Server
Using Local History	Exporting Actions
Using CVS	Importing Actions
Using SVN	Working with Mylyn Integration
Using FTP and SSH	Developing with JavaScript
Developing with Zend Framework	

Creating PHP Projects

PHP projects are the containers within which all PHP and other application files should be created.

To create a new PHP project:

1. From the menu bar, go to File | New | PHP Project

   -Or- In PHP Explorer view, right-click and select New | PHP Project.

   The New PHP Project wizard is displayed.

2. Enter the following information:
   - Project name - The required project name
   - Contents - Select whether to:
   - Create a new project in the workspace - Creates a new PHP project in the workspace directory.

     By default a workspace will have been created in

@user.home/Zend/workspaces/DefaultWorkspace7 when you first launched Zend Studio.

- Create a project from existing source - Creates a PHP project pointing to files situatued outside of the workspace.
  Click Browse to select the required source content.
- Create project on a local server - Creates the project on a local server. This option will only be available if a local Zend Server has been configured in the PHP Servers Preferences page.
- PHP Version - Select whether to:
  - Use default PHP settings - Uses the default PHP Interpreter settings.
  - Use project specific settings - Select the PHP version to be used for the project and whether ASP tags are used as PHP tags.
    See PHP Version Support for more information
- Project Layout - Select whether to:
  - Use project as source folder - All resources within the project will be added to the Build Path by default.
  - Create separate folders for source files and public resources - Separate folders will be created in which you can place resources which should be included or excluded from the Build Path.
    See Configuring a Project's PHP Build Path for more information.
    The default setting for this option can be configured from the New Project Layout Preferences page.
- JavaScript Support - Mark the 'Enable JavaScript support for this project' checkbox for JavaScript functionality (e.g. JavaScript Content Assist options) to be available to the project.
  See Enabling JavaScript Support in PHP Projects for more information.

3. Click Next to configure the project's Include Path (this can also be done following the project creation).
4. Click Next to configure the project's PHP Build Path (this can also be done following project creation).
5. Click Finish.

The new PHP project will be created in your workspace and displayed in PHP Explorer View. You can now start to develop your application by creating PHP Files or adding other resources to your project.

Creating PHP Files

PHP files can be created and opened in Zend Studio in a number of ways:

- Creating a new PHP file associated with a project
- Creating a new PHP file not associated with a project
- Opening an external file

Creating a PHP File within a Project

This procedure describes how to create a new PHP file within an existing project.

To create a new PHP file within a project:

1. In PHP Explorer view, select the Project within which you would like to place the file.
2. Right-click and select New | PHP File -or- go to File on the Menu Bar and select New | PHP File.
3. The PHP File creation dialog will be displayed.
4. Enter the name of the file and click Next.
5. The 'Use PHP Template' checkbox will be marked by default. This will create the new PHP file with the "<?php ?>" PHP tags.
   Select the required template or unmark the checkbox to create a blank file.
6. Click Finish.

Your file will open in the editor and will appear within your project in PHP Explorer and Navigator views.

Creating a PHP File Outside of a Project

Single PHP files can be created outside of a project quickly and easily for the purposes of writing short snippets of code which will not later need to be debugged or run and do not need to be associated with other files or projects.

To create a new PHP file not associated with a project:

1. From the Menu Bar, go to File | New | Other | PHP | Untitled PHP Document.
   -Or- click the new Easy PHP File icon on the toolbar .
2. A new PHP file, by default called PHPDocument1, will open in the editor.

Once the file has been created, it can later be saved within a project.

To save the file to a project:

1. Click save on the toolbar.
   A Save As dialog will open.
2. Select the project with which you would like to associate with the file
   -Or- To create a new project:
   i. Click Create New Project.
      The New PHP Project dialog will be displayed.
   ii. Enter the Project name and click Finish.
      The new project will be added to the list. Select it to save your file within your new project.
   iii. Edit the file name.
   iv. Click OK.

Your file will be saved within the selected project and will be available for running, debugging and profiling operations.

Opening an External File

These procedures describe how to open external files in Zend Studio.

External files can be opened in Zend Studio in three ways:

- Dragging-and dropping the file into Zend Studio.
- Double-clicking the file (Windows only).
- Using the Open function in Zend Studio.

Once external files have been opened in Zend Studio, you can perform operations such as running, debugging and profiling on them.

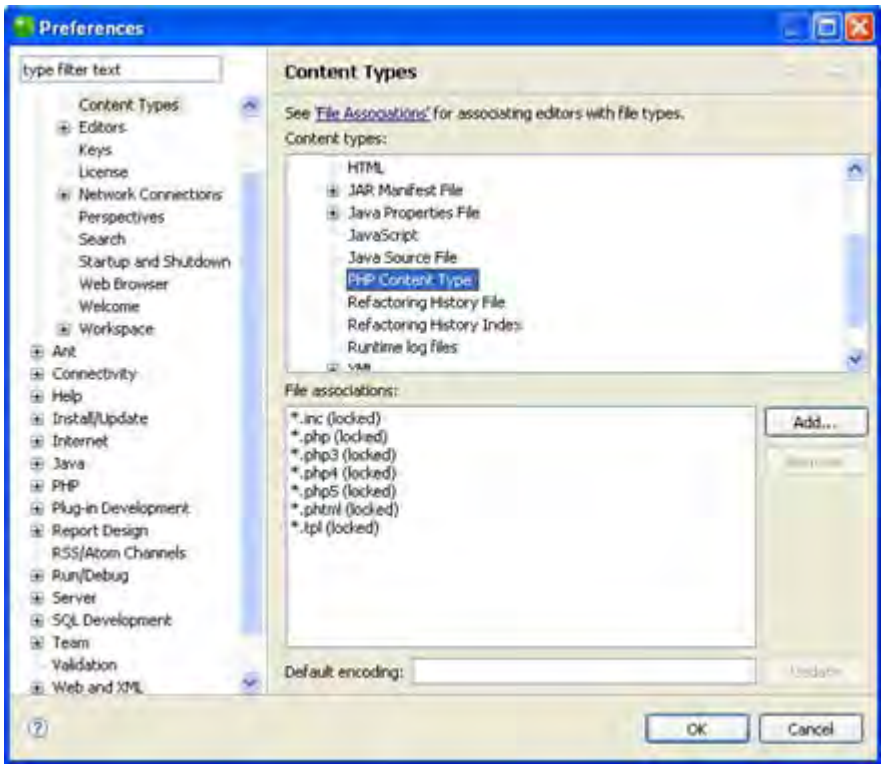
To open a file by dragging-and-dropping:

1. Find your file in your external file system.
2. Have both Zend Studio and your file system explorer open and visible on your desktop.
3. Drag and drop the file into the editor space in Zend Studio.

The file will be displayed in an editor and will be available for Zend Studio operations such as debugging and profiling.

To open a file by double-clicking:

1. If the file type you are trying to open was associated with Zend Studio during installation, simply double-clicking it in your external file system will cause it to be opened in Zend Studio.
2. If the file type was not associated with Zend Studio you can:
   - Right-click the file and select Open With | Choose Program | Zend Studio.
   - -Or- Add the file type to the list of file types which will automatically be opened in Zend Studio by doing the following:
     a. From Zend Studio's Menu Bar, go to Window | Preferences | General | Content Types.
        The Content Types dialog will be displayed.
     b. Select Text | PHP Content Type from the list.
        A list of file types associated with Zend Studio will be displayed.



   c. Click Add to add your file's type to the list,
   d. Enter the file type (e.g. .php) and click OK.
   e. The file type will be added to the list.
   f. Open your Windows Explorer.

g. Go to Tools | Folder Options | File Types.
h. From the File Types list, select PHP File.
i. In the Opens with category click Change, browse to your Zend Studio .exe location and click OK.
j. Click Apply.

You can now double-click the file on your external file system to open it in Zend Studio.

The file will be displayed in an editor.

To open a file using Zend Studio's file open function:

1. In Zend Studio, go to File | Open File.
2. Browse for your file in your file system.
3. Select the required file and click Open.

The file will be displayed in an editor.

Creating PHP Elements

New PHP Element wizards allow for the easy creation of PHP classes and interfaces. The wizards let you easily configure all required parameters and give you access to Superclasses.

New PHP Element wizards allow you to create a new PHP Class and creating a new PHP Interface.

Creating a New PHP Class

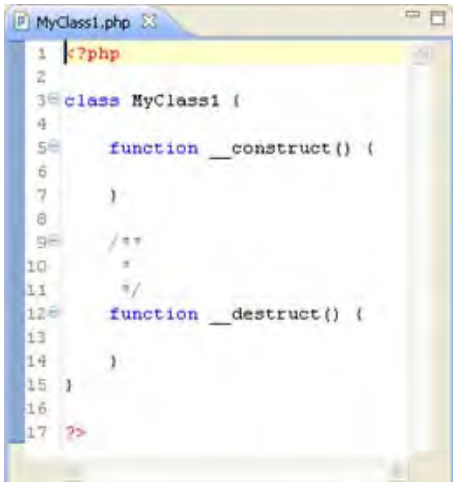
This procedure demonstrates how to create a new PHP class using the new PHP class wizard.

To create a new PHP class:

1. In PHP Explorer view, right-click the project/file in which you want to create the new class and select New | Class.
   The New PHP Class wizard is displayed.

2. Enter the following details:
   - Source Folder - If necessary, click Browse to change the soruce folder.
   - Class location - Select one of the following options:
     - Create New File - A new PHP file will be created in which the new class will be inserted
     - Add in existing file - The class will be created in an existing file.
       Click Browse to select the file in which it will be created and select whether it will be created as the 1st PHP Block in the file or as a New PHP Block at the end of the file.
   - Class Name - Enter the name for the class. If you chose the Create New File option, this will also be the name of the file.
   - Superclass - Click Browse to select a Superclass to extend.
   - Interfaces - Click Add to select interaces to extend/implement.
   - Method stubs - mark the checkboxes of the method stubs to be created (if any) from the following options:
   - Constructor
   - Destructor
   - Inherited abstract methods
   - Comments - mark the checkboxes of the comments to be created (if any) from the following options:
     - PHPDpc Blocks
     - TODOs
3. Click Finish.

The new class will be created with the required code.

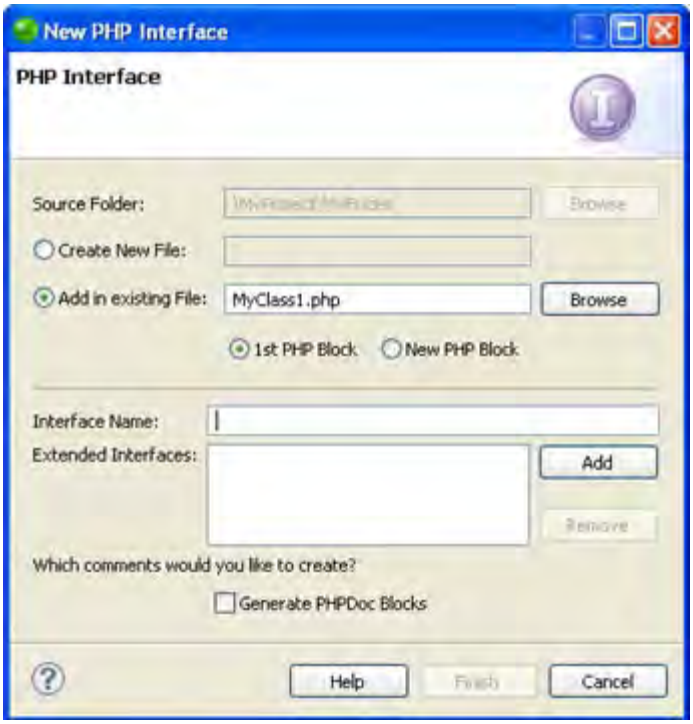


Creating a New PHP Interface

This procedure demonstrates how to create a new PHP interface using the new PHP Interface wizard.

To create a new PHP interface:

1. In PHP Explorer view, right-click the project/file in which you want to create the new interface and select New | Interface.
   The New PHP Interface wizard is displayed.

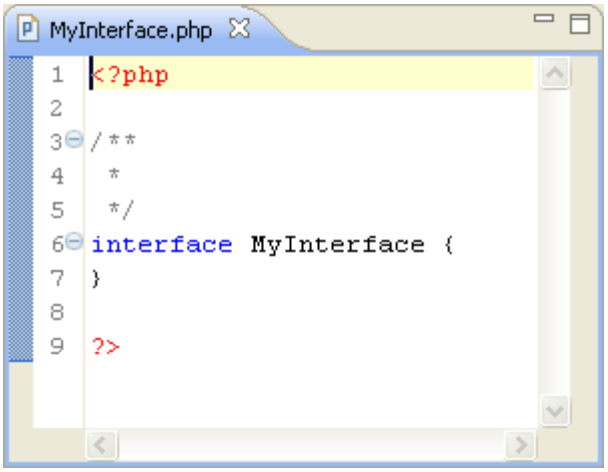


2. Enter the following details:
   - Source Folder - If necessary, click Browse to change the soruce folder.
   - Interface location - Select one of the following options:
     - Create New File - A new PHP file will be created in which the new interface will be insered
     - Add in existing file - The interface will be created in an existing file.
       Click Browse to select the file in which it will be created and select whether it will be created as the 1st PHP Block in the file or as a New PHP Block at the end of the file.
   - Interface Name - Enter the name for the interface. If you chose the Create New File option, this will also be the name of the file.

- Extended Interfaces - Click Add to select interfaces to extend/implement.
- Generate PHPDoc Blocks - Mark this checkbox for PHPDoc Blocks to be automatically created for this item.

3. Click Finish.

The new interface will be created with the required code.



Migrating From Zend Studio 5.X

If you have previously used Zend Studio, you can simply and easily migrate existing Studio 5.X projects and keymaps into Zend Studio 7.x:

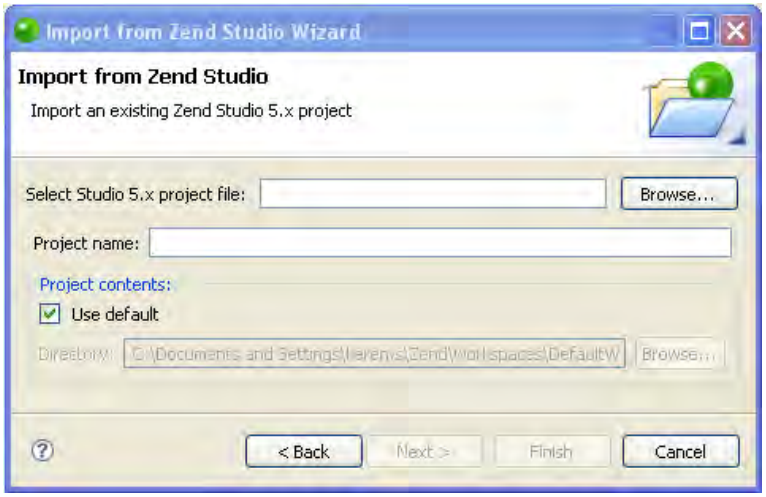
- Migrating Projects From Zend Studio
- Migrating Keymaps from Zend Studio

Migrating Projects From Zend Studio 5.X

This procedure describes how to quickly and easily import projects from Zend Studio 5.x into Zend Studio 7.x. Any links to CVS and SVN repositories will also automatically be created and maintained.

To import a project from Zend Studio5.x:

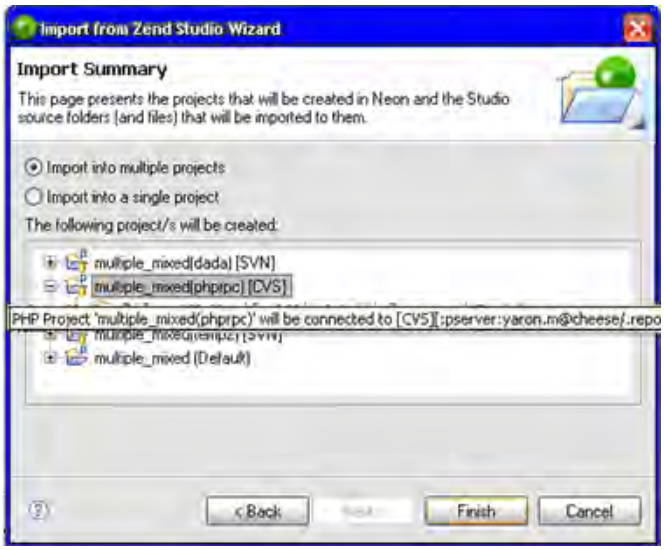
1. Right-click in PHP Explorer view and select Import | Zend Imports | Import from Zend Studio 5.x -or- go to File | Import | Zend Imports | Import from Zend Studio 5.x.
   The Import from Zend Studio Wizard will open.



2. Click Browse to find your Zend Studio project stored on your file system. The project will automatically have been stored with a .zpj file extension.
3. Click Open
4. The Project name will by default be given the same name as your .zpj file.
   Edit this name if required.
   If the project root contains folders linked to source control, the name given will be the project name specified in this screen with the name of the project root in parentheses following it. E.g. if the project name given was MyProject, and the

imported project root was MyProjectRoot, the Project will be displayed as MyProject (MyProjectRoot).

5. If you want the project to be imported into somewhere other than your current workspace, unmark the 'Use default' checkbox under the Project contents category and browse to a different location.
6. Click Next.
   The Import Summary dialog will open.


Import from Zend Studio Wizard
Import Summary
This page presents the projects that will be created in Neon and the Studio source folders (and files) that will be imported to them.
Import into multiple projects
Import into a single project
The following project/s will be created:
multiple_mixed(dada) [SVN]
multiple_mixed(phprpc) [CVS]
PHP Project 'multiple_mixed(phprpc)' will be connected to [CVS]:pserver:yaron.m@cheese/.repo
multiple_mixed (Default)
< Back
Finish
Cancel

7. If the project roots (folders and files) contained within your folder were not mapped to a Version Control System (CVS or SVN), they will all be contained within a single project in Zend Studio 7. This includes folders and files linked to FTP.
   If the project roots (folders and files) contained within your folder are mapped to a Version Control System, each project root will be imported into a separate project.
8. If you would like all projects root to be combined into one Zend Studio 7 project, select the 'Import into a single project' option. Note that in this case the Version Control System mappings will not be created.
9. Click Finish.

Your project(s) will be imported into Zend Studio 7 and will be available in PHP Explorer view.
The selected Version Control links will be maintained.
Any required FTP connections will be created in the Remote Systems View. See the FTP and SFTP Support topic for more on FTP/SFTP connectivity.

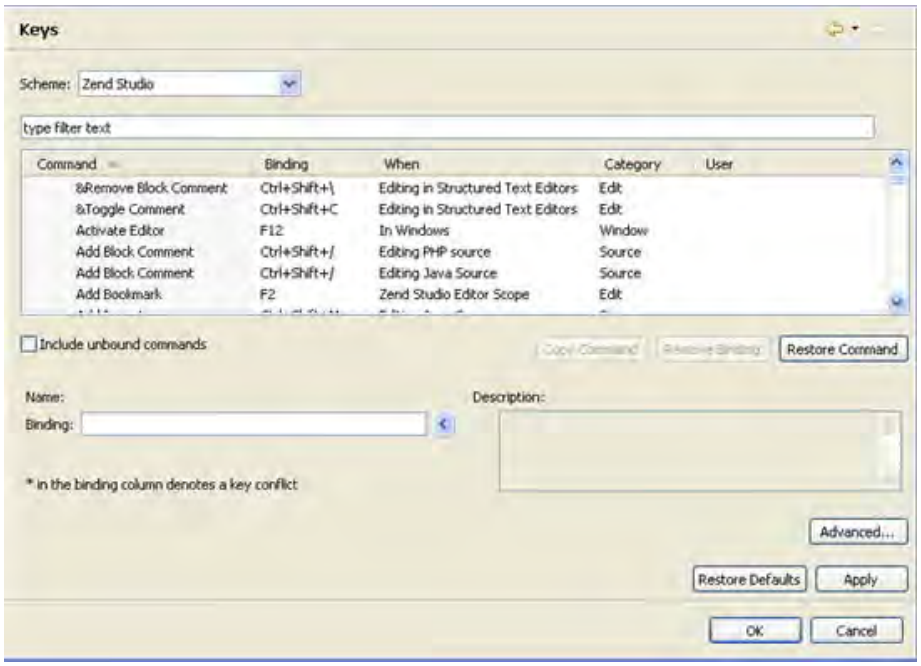
Note:
The project root itself needs to be mapped to a Version Control System so that its subfolders can be mapped.

Migrating Keymaps from Zend Studio 5.X

Zend Studio 5.X keymaps, or collection of command shortcuts, can be easily migrated and applied to Zend Studio 7.x for a seamless transition from Zend Studio 5.X to Zend Studio 7.x. This procedure describes how to apply a Zend Studio 5.X keymap so that Zend Studio 5.X's shortcuts will be available in Zend Studio 7.x.

To apply a Zend Studio 5.X keymap:

1. Go to Window | Preferences | General | Keys.
   The Keys preferences page will be displayed.



2. Select Zend Studio from the 'Scheme' drop-down list.
3. Click Apply.

Zend Studio 5.X's key shortcuts will be applied and available in Zend Studio 7.x.

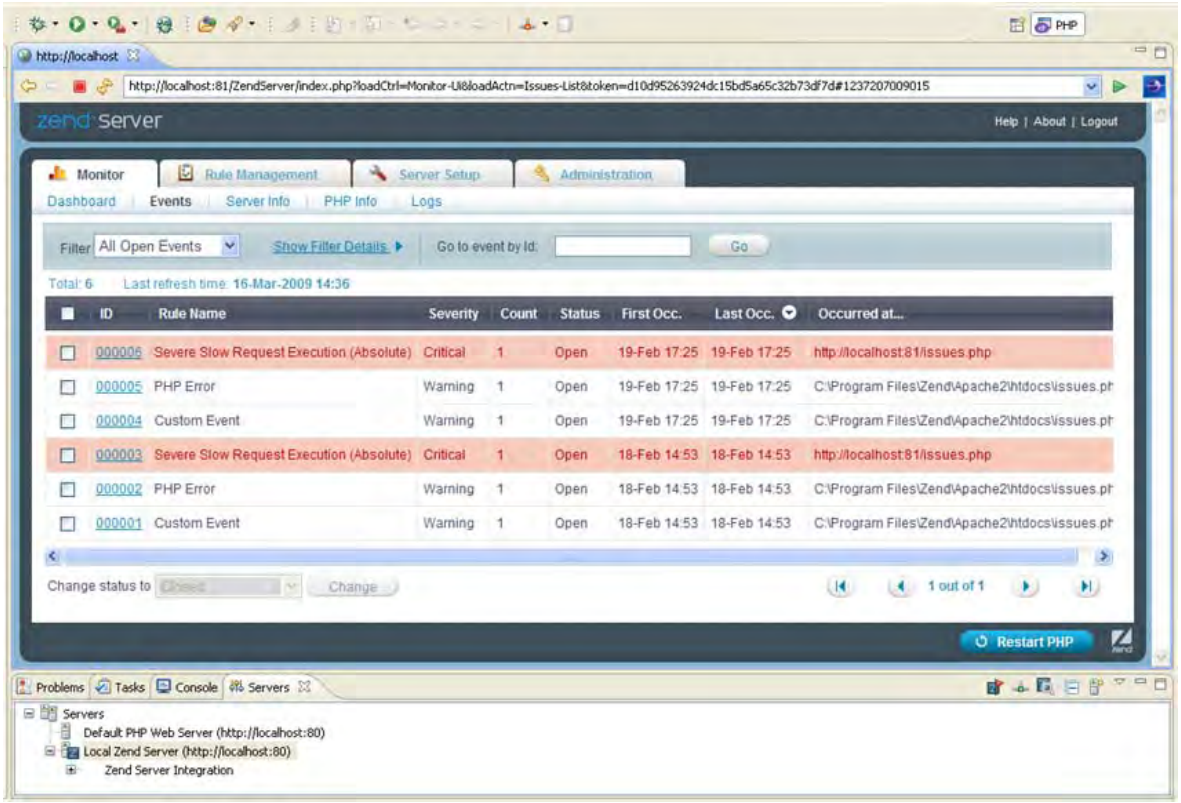
Zend Server Integration

Zend Server Integration

Integrating Zend Studio with Zend Server allows you to benefit both from Zend Studio's debugging and profiling functionality and from Zend Server's PHP Intelligence event monitoring capabilities.

Zend Server monitors and constantly tests your PHP environment and programs in order to allow you to gain maximum efficiency. Instances of problematic scripts and slow execution are captured by Zend Server as 'events.'
For more on events and how to configure what constitutes an event, see the Monitor chapter in Zend Server Online Documentation.

Zend Server's events can be viewed directly from Zend Studio using the internal browser, which displays the Zend Server User Interface.



Conversely, Zend Server's integration with Zend Studio means that the problems identified by Zend Server can be viewed, tested, debugged and profiled in Zend Studio.

Note:

If your Zend Server is situated behind a firewall or other security device, you will have to set up a tunnel in order for the integration between Zend Studio and Zend Server to be enabled. See Setting Up Tunneling for more information.

For more on Zend Server, go to http://www.zend.com/en/products/server.

Your Zend Server Integration might not function correctly for one of the following reasons:

- Your Zend Server is not correctly configured in Zend Studio.
  See Defining Zend Server in Studio for more information.
- Zend Server is situated behind a firewall or other security device.
  See Setting Up Tunneling for more information.
- Your Zend Studio Communication is not correctly configured in Zend Server.
  See Configuring Studio Communication Settings in Zend Server for more information.
- Zend Server is situated on an HTTPS Server. In this case, when trying to connect to your Zend Server from the Zend Server Events view, you will receive the following message:
  - For Windows: "Unable to verify the identity of 'http://<myserver>/ZendServer/index.php' as a trusted site."
  - For Linux: "Unable to verify the identity of 'http://<myserver>:10081/ZendServer/' as a trusted site."
    See Troubleshooting Platform Integration on an HTTPS Server for more information.

Working with Code Tracing

Working with Code Tracing

Zend Server Code Tracing captures full execution data (trace data) of PHP applications in real time. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure. Integrating Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. This feature is useful in resolving time performance issues, memory performance issues, and workflow errors. Using Code Tracing you can import a Zend Server event file and view the execution data which it contains in an editor.
Code Tracing uses the Zend Server Code Tracer Perspective to allow you to view the trace data within your environment.

The Code Tracing feature allows you to do the following:

- Import a Zend Server Event File
- Open the Source of Trace Data

Importing a Zend Server Event File

Importing Zend Server Event Files allows you to view and analyze trace data (execution data) from your server. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed files name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure.

Zend Studio allows you to import a .amf or .xml file:

- A .amf file is an event file which can be opened without access to Zend Server. The event file opens in the Zend Server Code Tracer Perspective and eliminates the need to recreate errors. Instead, you can locate errors directly in the trace data and then Open the Source of Trace Data. You may then edit the source inside of the corresponding local file.
- A .xml file is an event file which allows you to recreate an error when you have access to the Zend Server on which the event occurred. See Importing Events Into Zend Studio for more information.

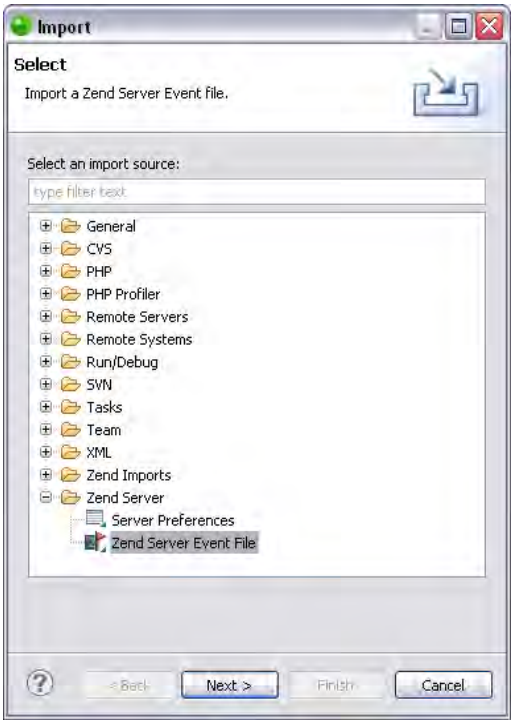
Note:

In order to import a Zend Server Event File, you must first export a Zend Server Event File from your server. For more information see Exporting Trace Information in the Zend Server Online Documentation.

Having an active connection with a licensed version of Zend Server or Zend Server Cluster Manager allows you to fully integrate the Code Tracing feature. For more information see Zend Server.

Importing a Zend Server Event File:

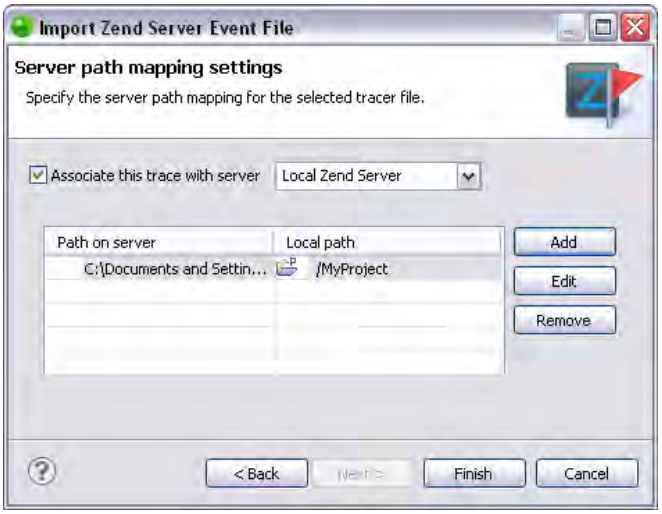
1. To open the Import Wizard go to **File | Import | Zend Server | Zend Server Event File** and click **Next**.



The "Import Zend Server Event File" dialog opens.

2. In the "From File" text field, browse to the location of your Zend Server Event File and click **Next**.

   The "Server path mapping setting" dialog opens.



3. Select the server to associate with from the "Associate this trace with server" drop down menu.
4. You can add, edit or remove a path map from this page using the appropriate buttons. You may select to have a path in your workspace or in the file system. For more information see Path Mapping.
5. Click **Finish** to save the changes.
6. Click **Yes** when the "Confirm Perspective Switch" message appears asking your permission to open the Zend Server Code Tracer perspective.
   The Zend Server Code Tracer Perspective opens with the imported Zend Server Event File open in the Tracer View.

For information about the functionalities available in the Zend Server Code Tracer perspective see Working with Code Tracing.

Opening the Trace Data Source

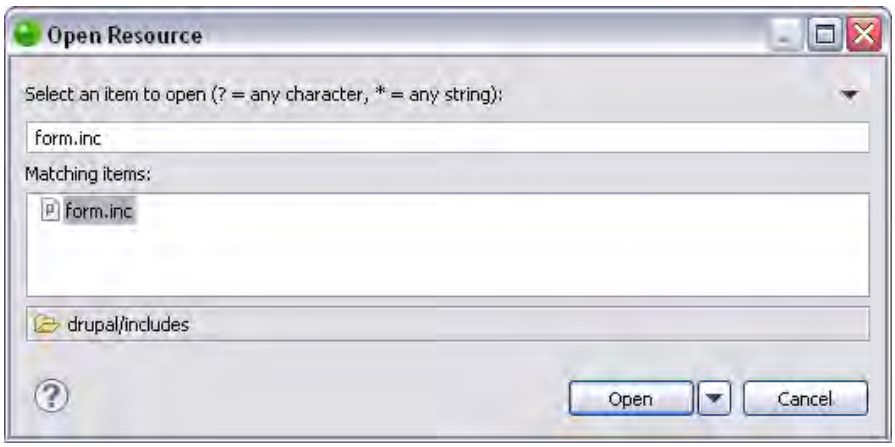
Opening the source of trace data (execution data) allows you to see the source of the functions inside Zend Studio. This functionality does not allow for editing of the source, but rather is used to pinpoint exactly where in the source the event is occurring. You may then edit the source inside of the corresponding local file.

Important Note:

To open the source of trace data you must first Import a Zend Server Event File for which the project already exists locally in your Zend Studio.

Opening the source of trace data:

1. To open the Zend Code Tracer perspective go to **Window | Open Perspective | Other | Zend Server Code Tracer.**
2. In the Tracer view go to the Statics per Function tab.
   A list of the functions appears. To see the memory usage of each function mark the "Show memory usage" option in the view.
3. Double click on a function.
   The "Open Resource" dialog opens.

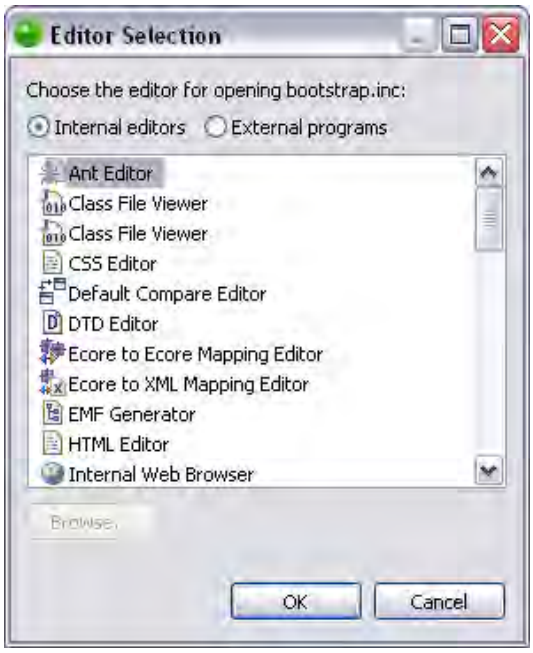


4. Use the drop down menu in the dialog to:

- Show the status line - Shows the status line in the current window.
- Show the derived resources - Resources that are not original data but can be recreated from their source files. For more information see Derived Resources in the Eclipse Online Documentation.
- Select/Deselect/Edit a working set - For information about working sets see Working Sets in the Workbench User Guide.

You can also use the **Open** button drop down menu to open the resource in a:

- PHP editor
- Text editor
- System editor
- In-Place editor
- Default editor
- Other - Opens the "Editor Selection" dialog and allows you to choose the editor (either an internal editor or an external program) to open your resource.



5. Select the resource and click **Open**.

   The source code of the resource opens in an editor view.

The source of the resource is read only. To edit a resource open the local project file in which it is located.

Debugging and Profiling Zend Server Events

Zend Server events can be debugged and profiled in Zend Studio either directly from Zend Server, using Zend Server integration, or by exporting an event from Zend Server and importing it into Zend Studio.

- Importing .xml Events Into Zend Studio
- Debugging Events from Zend Server

Importing .xml Events Into Zend Studio

Zend Server events can be exported from Zend Server into a .xml event file which allows you to recreate an error when you have access to the Zend Server on which the event occurred. The event can then be imported into Zend Studio and debugged.

For information about importing events using the Code Tracing feature see Working with Code Tracing. This will also allow you to import .amf event files into Zend Studio.

To debug a Zend Server event when you do not have direct access to the Zend Server:

1. Export the event(s) into a Zend Server event file.
2. Import and Debug the Zend Server event file in Zend Studio.

Creating a Zend Server event file

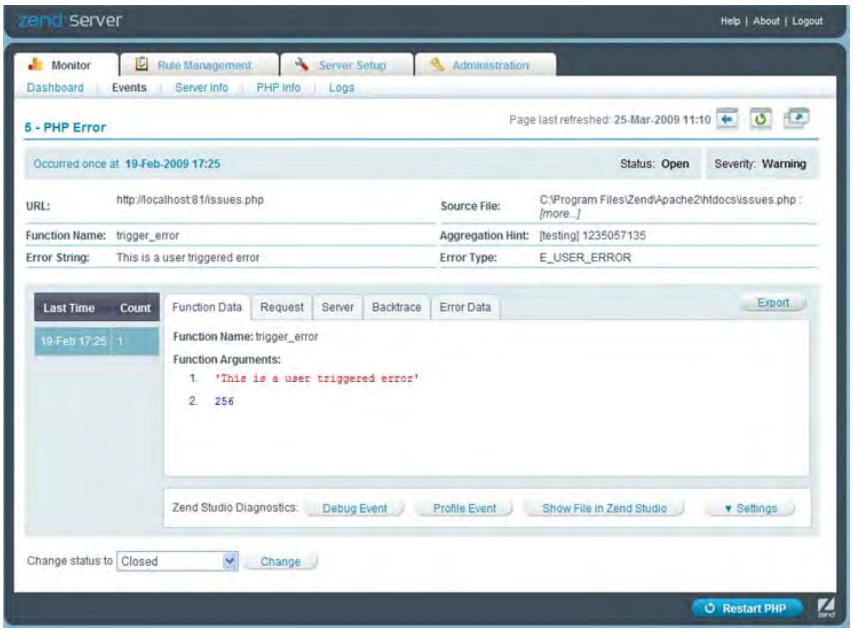
Note:
Zend Server events can only be exported through the Zend Server GUI.

To export an event from Zend Server:

1. Open your Zend Server GUI.

   Note:
   This can be done from within Zend Studio by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar . See Configuring Zend Server Settings in Zend Studio for more on configuring a Zend Server in Zend Studio.

2. Access the Event list by browsing to the Monitor | Events tab.
3. Click the Event which you want to export.



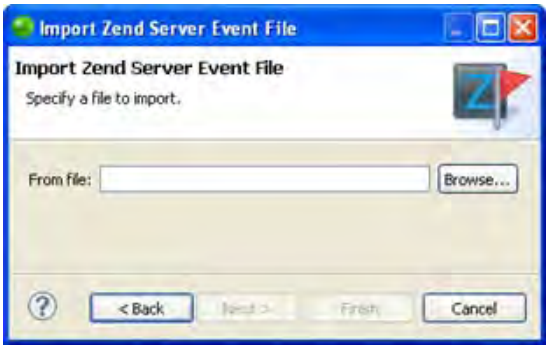
4. Click the Export button.
5. Select to save the file to a location on your file system.
6. The exported file will be created as an XML file containing the event details.

Once a Zend Server event file has been created, it can be imported and debugged in Zend Studio.

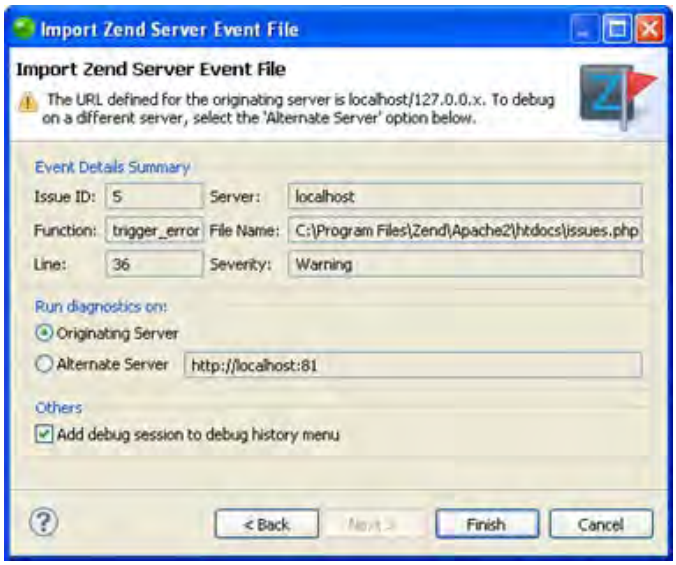
Importing and Debugging Zend Server Event Files

To import and debug a Zend Server event in Zend Studio:

1. From the menu bar, go to File | Import | Zend Server | Zend Server Event File and click Next.
   The Import Zend Server Event File dialog is displayed.



2. Click Browse and browse to the location of the event file.
3. Click Next.
   A dialog will display with the details of the event.



4. By default, the event will be debugged on the server according to the server address defined in the event file.
   To debug on a different server, select the Alternate Server option and enter the address for the alternate server.
   Ensure the server on which you are running the debug session is configured to run

remote debugging sessions. See Setting Up Remote Debugging for more information.

Note:

By default, a server defined on a local machine will have the server URL "localhost/127.0.0.x". If the address configured for the server from which the file was exported is configured as localhost, selecting the Originating Server option will run the debug session on the localhost server of the machine on which Zend Studio is located, which could be different than the server from which the event was exported. In this case, to debug the event on the originating server select the Alternate Server option and enter the URL of the originating server,

5. Mark the 'Add debug session to debug history menu' checkbox for the debug session to be available from the debug menu.
6. Click Finish.
7. The debug session will launch.
   See Running and Analyzing Debugger Results for more information on running a debug session.

Debugging / Profiling Events from Zend Server

Zend Studio allows integration with Zend Server so that Zend Server Events can be viewed, debugged and profiled in order to detect and locate errors and issues causing slow script execution.

Note

Before debugging Zend Server events, ensure the integration between Zend Studio and Zend Server is correctly configured as described in Setting Up Zend Server Integration.

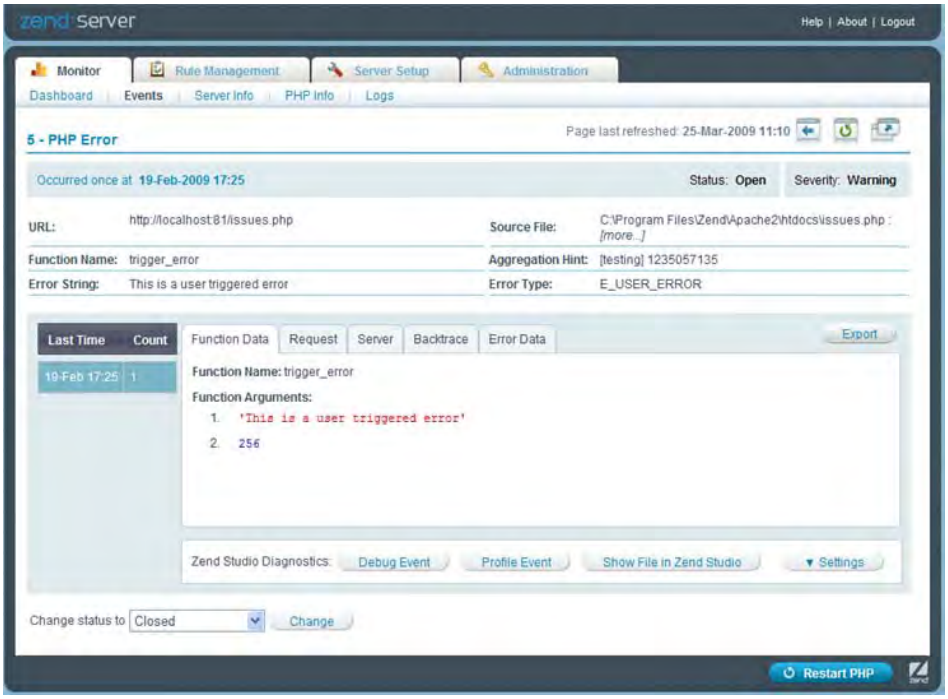
To debug an event directly from Zend Server:

1. Open your Zend Server Event List.

   Note:

   This can be done from within Zend Studio either through the Servers view or by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar . See Configuring Zend Server Settings in Zend Studio for more on configuring a Zend Server in Zend Studio.

2. Access the Event list by browsing to the Monitor | Events tab.
3. Click the Event which you want to debug.



4. To select the server on which the debug/profile session will be run, click the

button.
This gives you access to the following options:
   - Originating server - This will debug the event on the server from which the event originated
   - Alternate server - Allows you to debug the event on a different server (this server must also be running the Zend Debugger). Enter the IP address of the required server.
5. Click the

button to save your settings.
6. Click the

or

button.
7. The relevant debug / profile session is launched in Zend Studio.

Note:
If Zend Server cannot connect to Zend Studio, see both the see the Setting Up Zend Server Integration topic and the 'Error: Failed to Communicate with Zend Studio' topic in the Zend Server Online Help (http://files.zend.com/help/Zend-Serverzend-server/htm) for more information.

8. See Running and Analyzing Debugger Results for more information on running a debug session or the PHP Profile Perspective topic for more on the information displayed once a Profile session has been run.

Setting Up Zend Server Integration

In order to debug events through Zend Server, the integration between the Zend Server and your Zend Studio needs to be configured in both products.

See Configuring Zend Server Settings in Zend Studio and Configuring Zend Studio Settings in Zend Server for more information.

Note:

If you do not have access to the Zend Server on which the events were created, you can use the Import Zend Server Event function to import and debug a pre-created Zend Server event file.

Configuring Zend Server Settings in Studio

Zend Server settings can be configured in Zend Studio in order to allow the appliance of Zend Studio functionality (Profiling, Debugging etc.) to Zend Server Events, as well as allowing access to Zend Server's Event list.

Zend Server configuration can be done either automatically, if the Zend Server is installed on the same machine on which Zend Studio is running, or manually through the PHP Servers Preferences page.

Automatic Zend Server configuration

Zend Servers installed on the same machine as Zend Studio are automatically detected and configured in Zend Studio.

To automatically configure a Zend Server:

The auto detection is triggered when Zend Studio is launched or when the Auto Detect Zend Server button is clicked

For auto detection when Zend Studio is launched:

1. Ensure Zend Server is installed and running on the local machine.
2. Open Zend Studio.

   A popup ballon will appear in the bottom-right corner of the window indicating that a Zend Server installation has been detected and configured.

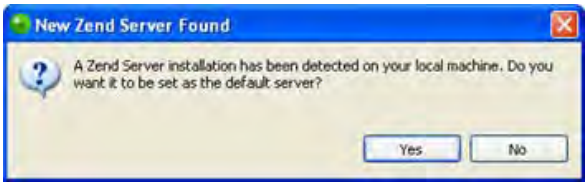
Zend Server Found
A Zend Server installation has been detected on your local machine and added as your default PHP server.
Click here to learn more

3. Click the icon to close the balloon or the 'click here' link to be taken to the Zend Server Integration help page.

To auto detect Zend Server after Zend Studio has been launched

You can manually trigger a Zend Server auto detection through the Servers view by performing the following steps:

1. Open the Servers view.
   This is available from the default PHP Perspective or can be manually opened by going to Window | Show View | Zend Servers | Servers.
2. Click the Auto Detect Zend Server icon on the toolbar.
3. If a Zend Server installation has been detected, a New Zend Server Found dialog is displayed asking whether you want to configure it to be the default server.


New Zend Server Found
A Zend Server installation has been detected on your local machine. Do you want it to be set as the default server?
Yes
No

4. Click Yes to set it as the default server or No for the server to be configured by but not set as the default.

A Local Zend Server configuration is confiured and added to your PHP Servers Preferences page.

To configure the automatically detected Zend Server:

In order to access the Zend Server GUI and Zend Server Event List from Zend Studio, you will need to configure the Zend Server GUI password for the newly configured Zend Server by performing the following steps:

1. Open the PHP Servers preferences page by going toWindow | Preferences | PHP | PHP Servers.
2. Select the Zend Server from the list (by default, it will have been named Local Zend Server).
3. Click Edit.
4. Go to the Zend Server tab.
5. In the Authentication category, enter the password used to access the Zend Server GUI.
6. Click OK and Apply.

You will now be able to access the Zend Server Event list by selecting the Zend Server from the the drop-down list next to the Zend Server icon on the toolbar or by clicking the Show Server Event List icon in the Servers View.

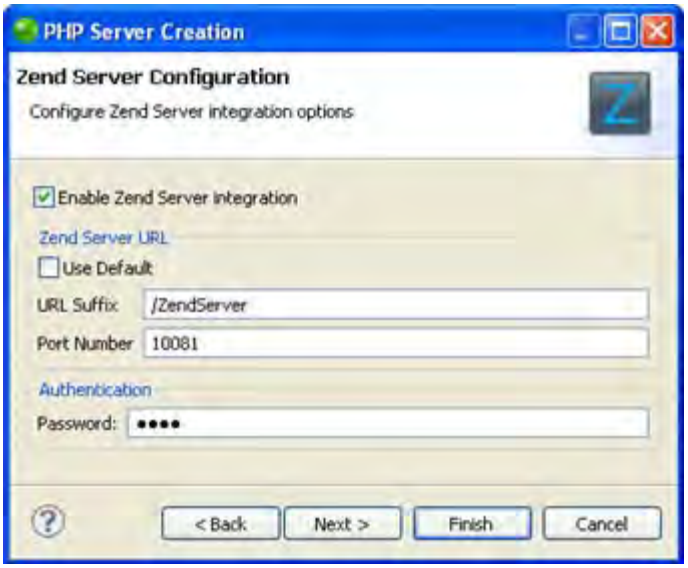
Manual Zend Server Configuration

To manually configure a Zend Server:

1. Open the PHP Servers Preferences page by going to Window | Preferences | PHP | PHP Servers.
2. Click New to create a new server with Server Integration.
   A PHP Server Creation dialog will open.
3. Configure the server as described in the PHP Server Preferences page (enter the server's name and the URL of your document root.)

   Note:
   If you are unsure of your Zend Server's document root, see the Zend Server FAQ to find out what your docunent root is.

4. Click Next.
5. If necessary, define Path Mapping. See Adding a Server Location Path Map for more information.
6. Click Next.
   The Zend Server Configuration dialog is displayed.



Zend Server Configuration dialog

7. Mark the Enable Zend Server integration checkbox to enable Zend Server integration functionality.
8. Leaving the Use default checkbox marked will create a URL in the format <server's document root>/ZendServer>. If necessary, unmark the checkbox and edit the following:
   - URL Suffix - The suffix which should be added to the URL of your document root in order to browse to your Zend Server GUI.
   - Port number - Enter the port number you defined during Zend Server installation. See the Zend Server FAQ site for default port number settings.
9. Enter the password for your Zend Server GUI.
10. Click Next to configure Tunneling settings or Finish to create your server.

Your Zend Server is added to the Zend Server list and will allow you to use Zend Server integration features.

Your Zend Server is now available from the arrow next to the Zend Server icon on the toolbar.

Configuring Zend Studio in Zend Server

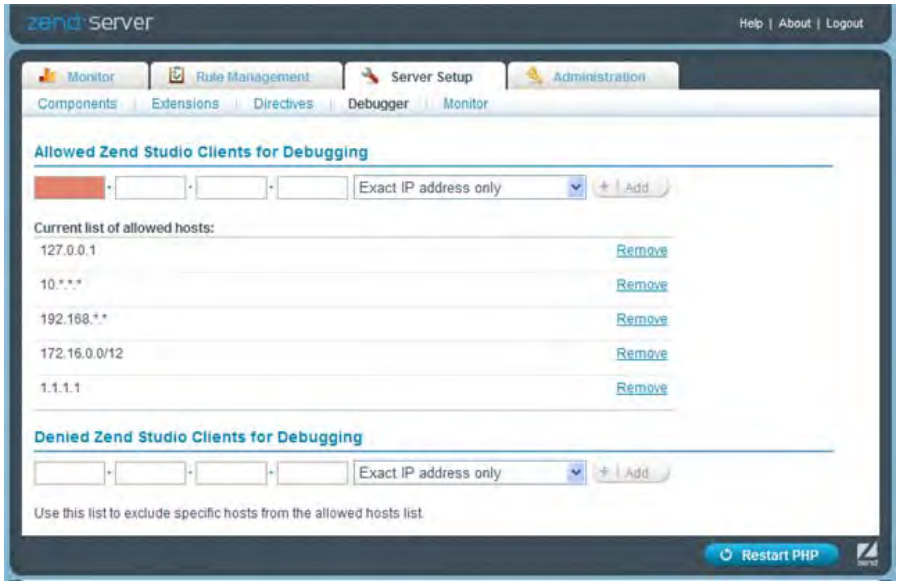
In order for you to be able to debug applications situated on Zend Server, your Zend Studio must be configured as an Allowed Host for initiating debugging settings.

To configure Zend Studio communication in Zend Server:

1. Open your Zend Server GUI.

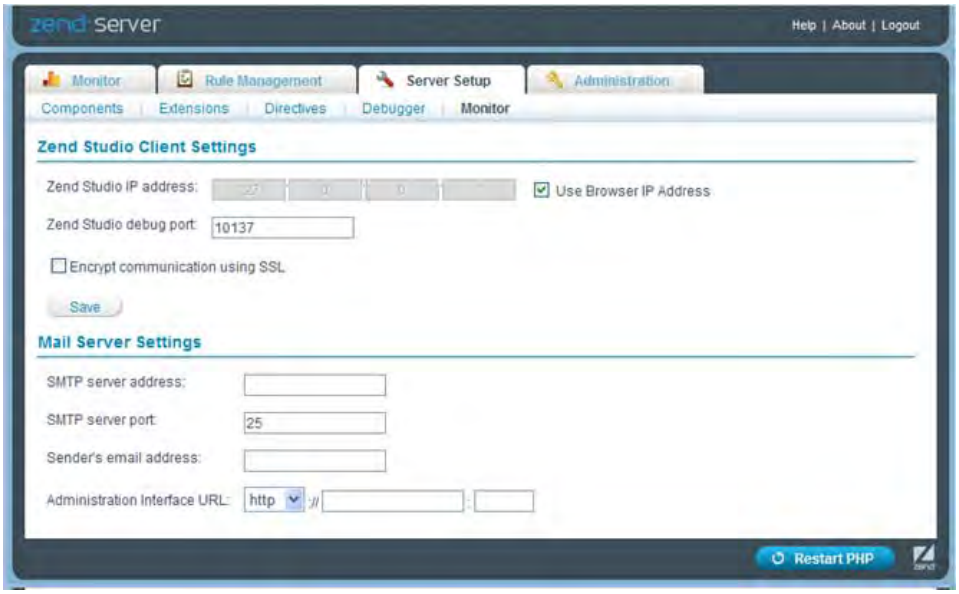
Note:
This can be done from within Zend Studio through the Servers view or by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar .

2. Go to the Server Setup | Debugger tab.



3. Ensure the IP address of your Zend Studio is included in the Allowed Hosts sections.
   To add an address to the list:
   i. Under the Allowed Zend Studio Clients for Debugging category, enter the IP of the machine on which your Zend Studio is installed
   ii. Click Add.
   The IP Address is added to the Allowed Hosts list.
4. Ensure your Studio's IP address is not in the Denied Hosts list.
   If it is, click Remove next to the required address to remove it from the list.
5. Click the Restart PHP button to apply your settings.

6. Browse to the Server Setup | Monitor tab.



7. Configure the following settings under the Zend Studio Client Settings category:
   - Zend Studio IP address - Enter the IP address of the machine on which your Zend Studio is installed or mark the Use Browser IP address to use the IP address of the machine on which the Zend Server UI is running.
   - Zend Studio Debug Port - This should match the debug port configured for the Zend Debugger in Zend Studio's Installed Debugger Preferences page (Window | Preferences | PHP | Debug | Installed Debuggers) page. The default port number is 10137.
   - Encrypt communication using SSL - mark this checkbox for the communication between Zend Studio and Zend Server to be encrypted using SSL.
8. Click the

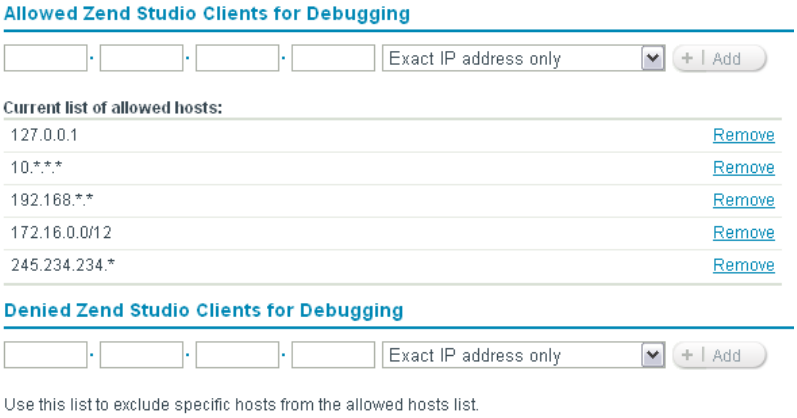
button to apply your settings.

Configuring Studio Communication Settings in Zend Server

In order for you to be able to view, debug and profile Zend Server Events in Zend Studio, you must ensure the correct communication settings are configured in your Zend Server.

To configure Zend Studio communication in Zend Server:

1. Open your Zend Server GUI.
2. Go to the Server Setup | Debugger tab.



3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections. This will ensure you can debug/profile Zend Server Events.
   To add an address to the list:
   i. Enter the IP address or Net mask of the machine on which your Zend Studio is installed. In Order to enter a Net mask, enter a range by entering the beginning of an IP address and adding '0' instead of the rest of the number. To make sure you are using Wildcards (*) to specify a range of IPs select the pattern you want from the drop-down list.
   ii. Click Add. Your Zend Studio machine's address will be added to the Allowed Hosts list.
4. Ensure your Zend Studio's IP address is not in the Denied Hosts list.
   If it is, click Remove next to the required address to remove it from the list.
5. In the Zend ServerGUI, go to the Server Setup tab | Monitor and configure the following:
   - Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to automatically detect your Zend Studio Debug settings.

6. Click Save.
7. Restart your Web server for the settings to take effect.

Configuring Zend Server to Auto Detect Studio Settings

This procedure describes how to configure Zend Server so that Zend Studio's settings are automatically detected during the Debugging / Profiling of Zend Server events.

To establish a communication tunnel between Zend Server and Zend Studio:

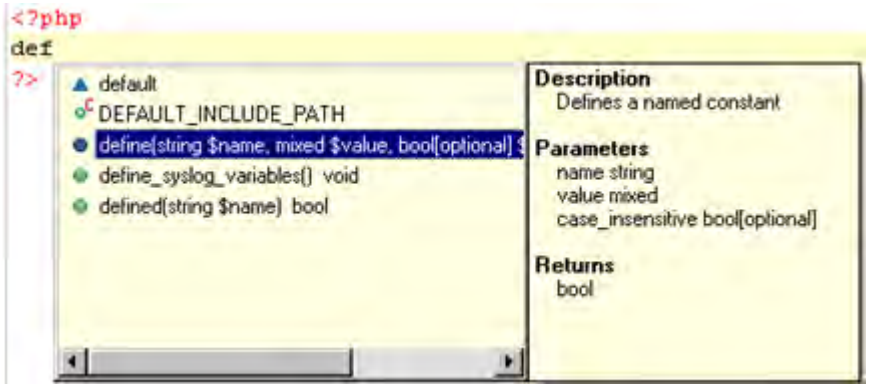
1. Open your Zend Server GUI.
2. Go to the Server Setup | Monitor tab.
3. In the Zend Server Settings section, configure the following:
   - Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to detect your Zend Studio Debug settings.
4. Click Save.

Using Content Assist

This procedure describes how to use Content Assist in order to quickly and easily insert code elements into your script:

To use Content Assist:

1. Enter the first few characters of the required code element into the editor.
2. The Content Assist window should be automatically displayed.
3. If the Content Assist window does not pop up automatically, press Ctrl+Space.



3. Use the arrow keys to scroll through the code completion options. The window on the right will display descriptions and hints for using the selected code element.
4. Select the required option by double-clicking it or selecting it and pressing Enter.

The selected code will be inserted into your script.

To enable the Content Assist window to open automatically, go to the Content Assist Preferences page, accessed from **Window | Preferences | PHP | Editor | Content Assist** and mark the "Enable auto-activation" checkbox. This is marked by default.

Using Templates

Templates are shortcuts used to insert a pre-defined framework of code into your scripts. The purpose is to save time and reduce the potential for errors in standard, repetitive code units. Once a template is inserted, you can complete the code quickly using manual and automated code entry methods.

Requirements:

A template must be defined in the Templates list in the Template Preferences page, accessed from Windows | Preferences | Templates, before it can be used. To learn how to create a new template from the Templates Preferences page, see Adding a New Template.

Note:

The i5 edition of Zend Studio includes pre-defined templates for using i5 PHP API Toolkit functions. See i5 Edition Extras for more information.

Templates are context sensitive and can be used in HTML, PHP, PHPDOC, JavaScript or CSS. The context of the current code being entered defines which templates are available. For example, PHP templates are not available if your current code is Java.

Inserting a Template into Code

This procedure describes how to insert a template into your script.

To insert a template:

1. Place your cursor at the desired insertion point.
2. Enter a character string (e.g. "Sw").
3. Click Ctrl+Space. The Content Assist box will appear, listing all available templates and completion options that begin with that combination of keys.

   Templates are marked in the content assist list with a blue square.
4. Double-click the required template from the list.

The template will be entered into your code.

Example:

Entering "sw" and selecting the the "switch statement" template from the list will give you the following code:

```
switch ($var) {
    case value:
    ;
    break;

    default:
        ;
    break;
}
sw
```

Note:

Templates can be created, imported and exported through the Template Preferences page, accessed from Window | Preferences | PHP | Templates.

Formatting Code

Zend Studio can auto-format your code according to set standards in order to make it easily navigable and readable.

Your code will be automatically formatted according to the settings defined in the PHP Formatter Preferences page, accessed from Window | Preferences | PHP Formatter.

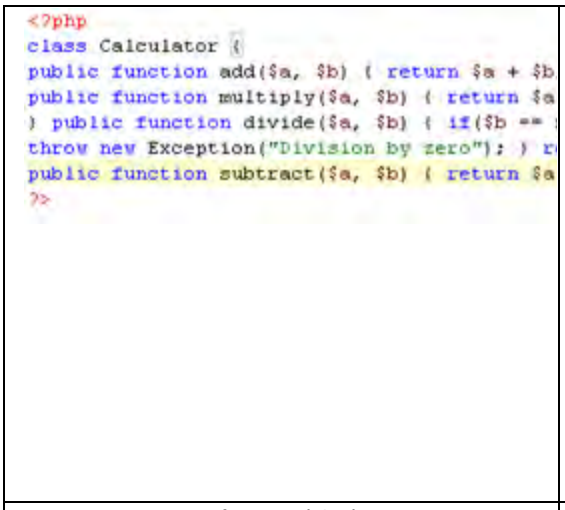
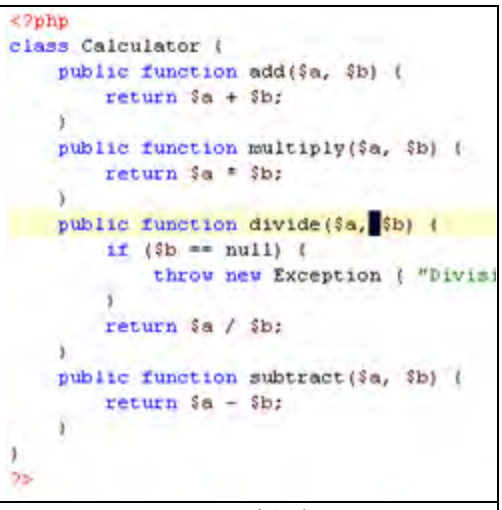
This procedure demonstrates how to format your scripts.

To format your whole script:

1. Open the required file.
2. Go to Source | Format Document or press Ctrl+Shift+F.

Your code will be automatically formatted.

Example:

Unformatted Code	Formatted Code
	

To format only selected lines within the script:

1. Select the relevant lines.
2. Go to Source | Format Active Elements -or- press Ctrl+I.

Only the selected lines will be formatted.

Note:

Code Formatting will also be available in JavaScript editors.

JavaScript Formatting preferences can be configured from Window | Preferences | Web | JavaScript | Code Style | Formatter.

Using Code Folding

Code Folding collapses or "folds" the display of a block of code.

To enable code folding, go to the Folding Preferences page, accessible from Window | Preferences | PHP | Editor | Using Code Folding.

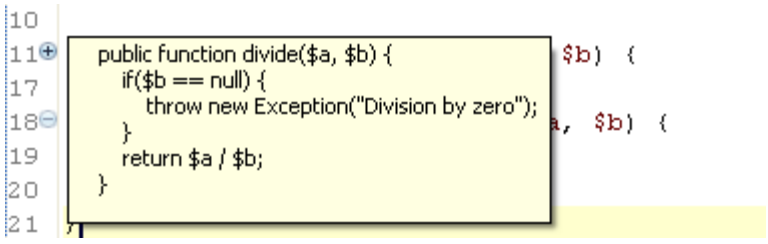
If Code Folding is enabled, minus signs will appear in the Annotation Bar next to code blocks which can be folded. In addition, certain elements will be folded by default according to the Folding Preferences settings.

To fold a block of code:

1. Stand within a class, function or PHPDocBlock.
2. Click the minus sign on the marker bar to the left of the editor.

The first line of code will remain visible but the other lines will not be displayed. A fold indicator ... will appear at the end of the line when the code is folded to indicate that there is hidden code.

To temporarily view folded code, hover over the plus sign that denotes a folded block. The folded code will be displayed in a floating block.

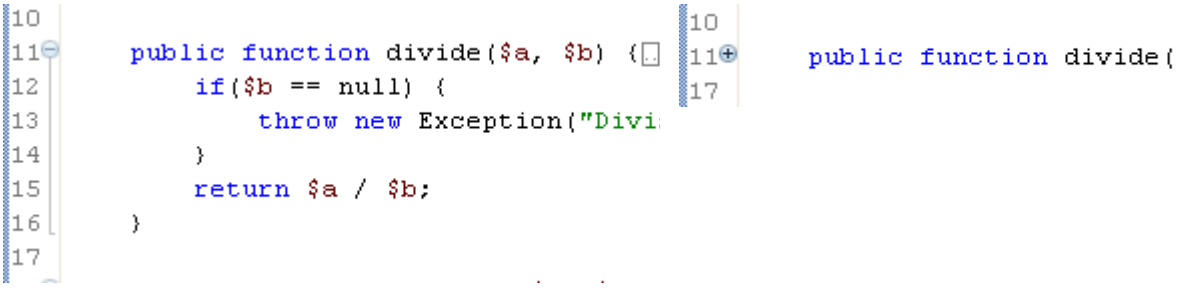


To unfold a block of code:

1. Click the plus sign.
2. The folded code will become visible again and the fold indicator will disappear.

To view the scope of a fold:

1. Hover over the minus sign.
2. A vertical line will be displayed from the first to the last line of the fold, indicating its range.

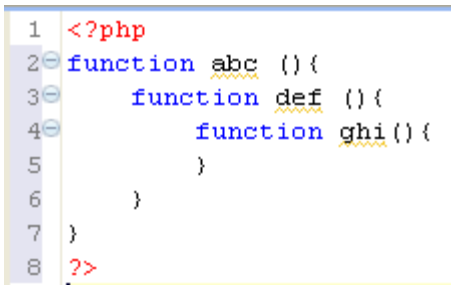


An unfolded function

A folded function

To fold/unfold nested functions:

1. Click on one of the minus signs of a nested function. All levels below this level will be folded into it. You can continue to fold until all levels have been folded into the topmost level.
2. To unfold nested functions, click on the plus sign. The folded code will open in the same order that it was folded.



Nested Function Folding

Note:

Line numbers are folded together with the code. Folding and unfolding does not change line numbers, it can only hide / display them.

Note:

If the folded code contains an error, the displayed window will be syntax highlighted on both the left and right Annotation bars.

Searching for PHP Elements

Searching for PHP elements is a functionality that allows you to search for PHP elements with your defined specifications. PHP elements are classes, functions and constants that are used in your code. Use this option if you want to locate one of these elements in your workspace, project or in a single file.

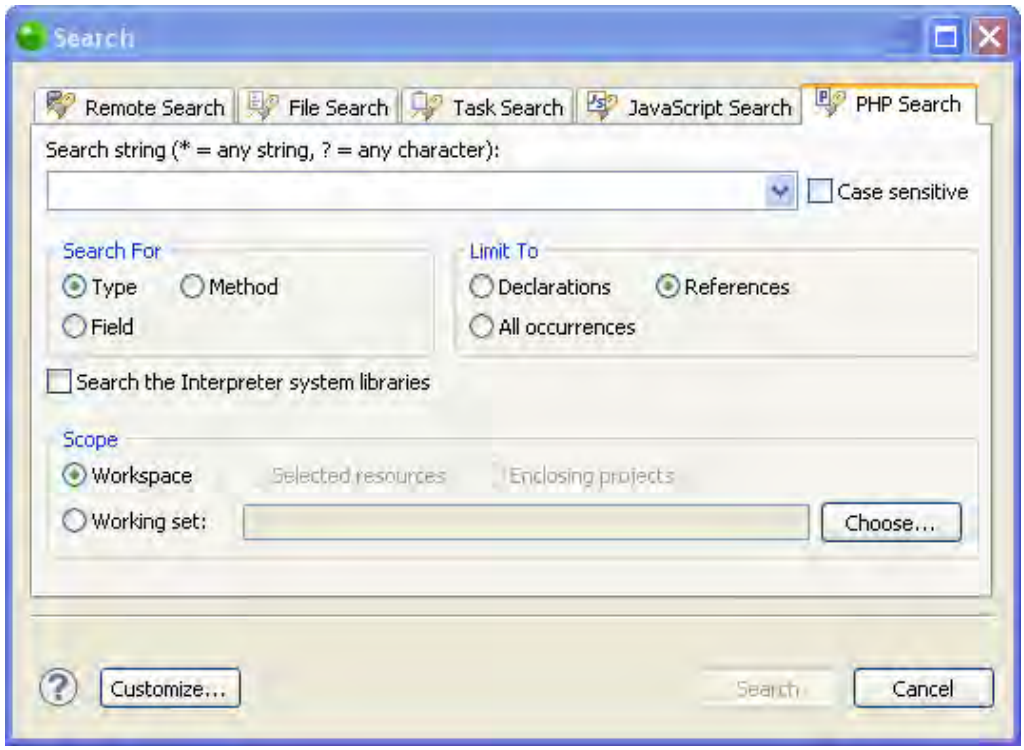
Note:

For information about the "Search" tab in the "Search" dialog, read below. To learn more about the additional tabs in the "Search" dialog see File Search in the Workbench User Guide. If you are using the Remote Search option, be aware that it does not search all of the resources of the remote server.

This procedure describes how to search for PHP elements (classes, functions and constants) within your files and projects.

To search for a PHP element:

1. From the Menu Bar go to **Search | Search** -or- press Ctrl+H.
   The "PHP Search" dialog will open.



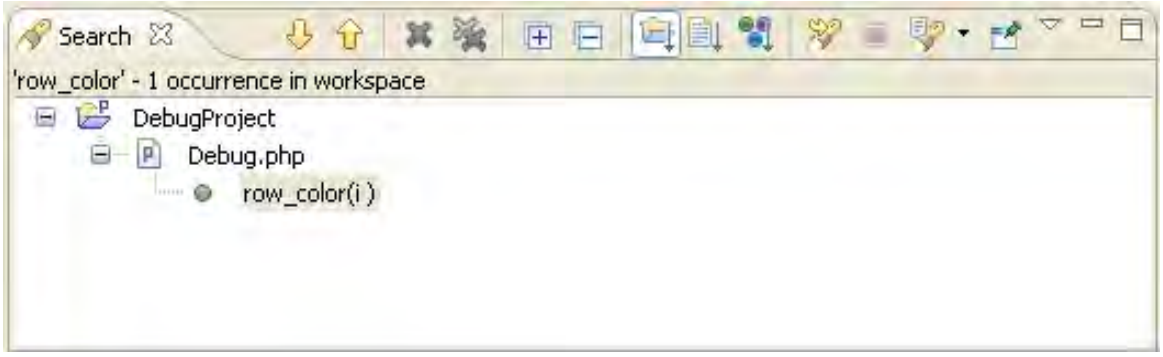
PHP Search dialog

2. Enter a search string. You have the option of using Wildcards.

You can make your search case sensitive by selecting the "Case sensitive" box.

3. Select whether to search for a:
   - **Type** - A class (This option will search for both methods and fields.)
   - **Method** - A function
   - **Field** - A constant
4. Select whether to limit your search to:
   - **Declarations** - The location where the type, method, or field are defined.
   - **References** - Anywhere the type, method, or field are referred to.
   - **All occurrences** - Anywhere the type, method, or field are referred to or declared.
5. To extend your search to include interpreter system libraries, select the "Search the Interpreter system libraries" box. This allows Zend Studio to search in the libraries of the PHP version you have selected in the PHP Interpreter Preferences.
6. Select whether to search in:
   - **Workspace** - The entire workspace
   - **Selected resources** - Select these in PHP Explorer view before opening the "Search" dialog. All sub-files and sub-folders within the selected resource will be included in the search.
   - **Enclosing projects** - The projects which the selected resources are in.
   - **Working Set** - Click **Choose...** to select the required Working Set.
7. Click **Search**

The search view will open displaying the results of the search.
To go to an element, double-click the required option from the search view.



Note:

By default, the File Search, Remote Search, Task Search, and JavaScript Search dialogs will be tabbed with the PHP Search dialog. To make these unavailable, click **Customize...** within the PHP Search dialog and unmark required options.

Hotkeys

The following hotkeys are available:

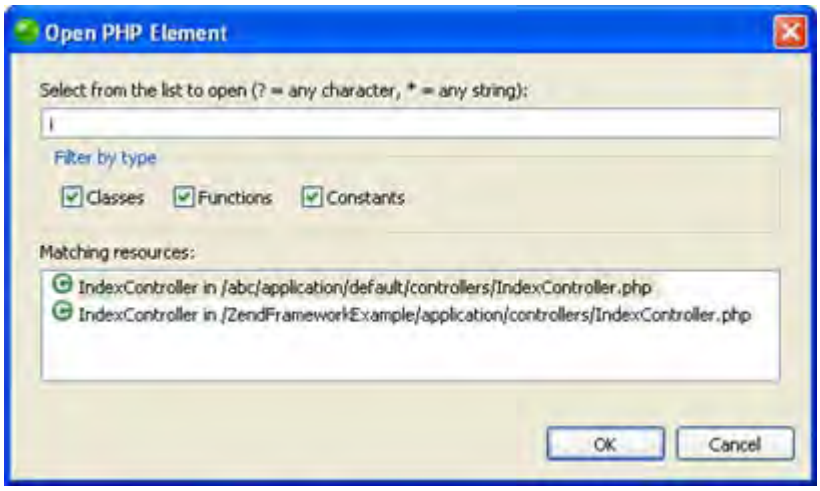
Action	Shortcut
Esc / Alt+F4	Closes the Search dialog.
Alt + A	Brings the cursor to the "Search string" field.
Alt + I	Checks/unchecks the "Case Sensitive" checkbox.
Alt + T	Checks "Type" in the "Search For" box.
Alt +M	Checks "Method" in the "Search For" box.
Alt +F	Checks "Field" in the "Search For" field.
Alt +L	Checks "Declerations" in the "Limit To" field.
Alt +R	Checks "References" in the "Limit To" field.
Alt +O	Checks "All occurrences" in the "Limit To" field.
Alt +Y	Checks/unchecks the "Search the Interpreter system libraries"
Alt + W	Selects "Workspace" in the "Scope" field.
Alt + D	Selects "Selected resources" in the "Scope" field.
Alt + J	Selects "Enclosing projects" in the "Scope" field.
Alt + K	Selects "Working set" in the "Scope" field.
Alt + H	Opens the "Select Working Sets" dialog.
Alt + S	Performs the search and shows the results in the Search View.
Alt +Z	Opens the "Search Page Selection" dialog and allows you to customize the search page.

Opening PHP Elements

This procedure describes how to use the Open PHP Element function to navigate to a PHP element (Class, Function or Constant) in an open project.

To open a PHP Element:

1. From the Menu Bar, go to Navigate | Open PHP Element -or- click the Open PHP Element icon on the Toolbar .
   The Open PHP Element dialog will open.
2. Enter the first few characters of the element which you want to open.
   Resources that begin with those letters will appear in the 'Matching Resources' pane, listed alphabetically.



3. You can filter by element type (class, function or constant) by marking/unmarking the relevant checkboxes.
4. Select the required element and click OK.

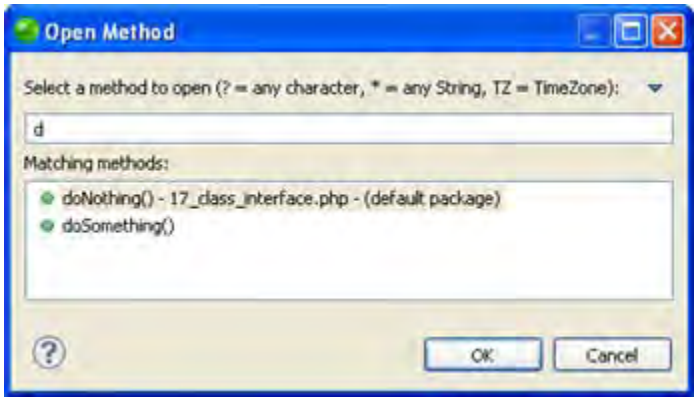
The file containing the element declaration will open in the editor, with the element highlighted.

Opening Types/Methods

You can open any method or type in your workspace using the Open Type wizard or the Open Method wizard.

To open a Type or Method:

1. From the Menu Bar, go to Navigate | Open Type or Open Method
   -Or- press Ctrl+Shift+T (for a type) or Ctrl+Shift+M (for a method).
   The Open Type/Method dialog will appear. If a type/method was previously selected in the editor or outline views, it will be displayed.



2. Begin typing the string of the required type/method to filter the results.
   You may use wild cards or CamelCase notation (e.g. DTB for DateTextBox).
3. Select the required type/method from the list and click OK.
   You may restrict the results displayed in the Open Type list to a particular Working Set through the wizard's menu (accessed by clicking the arrow in the top-right corner).

An editor will open on the selected type/method.

Note:

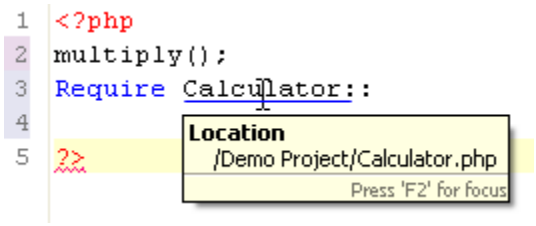
The Open Type dialog maintains a history of recently opened types. These are shown when the dialog is opened and stay above a separator line when you start to type a filter expression.

Using Smart Goto Source

This procedure describes how to use the Smart Goto Source function in order to easily navigate to an element's declaration.

To use the Smart Goto Source function:

1. Hover over the element whose source declaration you want to navigate to.
   A tooltip will be displayed showing the element's original location.



Element with its source location

2. Hold down the Ctrl key and move the cursor until the element is underlined.
3. Click the element.

You will be automatically taken to the element's source code. If the declaration is in a different file, this file will be opened.

Note:

Smart Goto Source will also be available for JavaScript elements if JavaScript support was enabled for the project. See Enabling JavaScript Support in PHP Projects for more information.

Viewing Type Hierarchies

Type hierarchies can be viewed in either a Quick Type Hierarchy view or in the Type Hierarchy view.

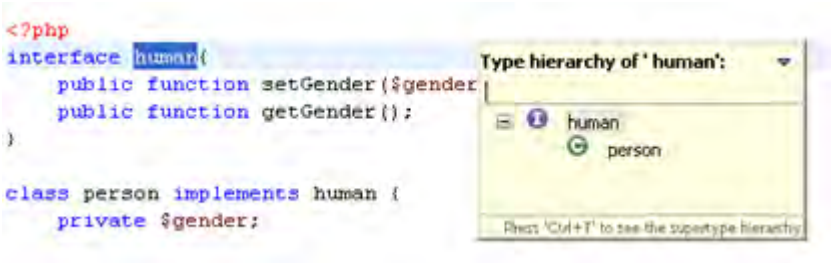
Viewing Types in the Quick Type Hierarchy View

To view a type in a Quick Type Hierarchy view:

1. Select the type name in the PHP editor, or PHP Explorer or outline views.
2. Press Ctrl+T

   -Or- from the Menu Bar go to Navigate | Quick Type Hierarchy.

The Quick Type Hierarchy view will be displayed in the editor with the selected type.



Note:

Pressing Ctrl+T while the quick type hierarchy view is shown will toggle between supertype hierarchy and subtype hierarchy.

Viewing Types in the Type Hierarchy View

Types can be viewed in the Type Hierarchy view by searching for them using the Open Type in Hierarchy dialog or by directly selecting the element in the editor or PHP Explorer.

To view a type in the Type Hierarchy view:

Through the Open Type in Hierarchy wizard:

1. Press Ctrl+Shift+H -or- from the Menu Bar go to Navigate | Open Type in Hierarchy.

   The Open Type in Hierarchy dialog is displayed.

2. If a type was previously selected in the editor or outline views, it will be entered in the type field.
   If a type was not selected, begin typing the string of the required type/method to filter the results.
   You may use wild cards or CamelCase notation (e.g. DTB for DateTextBox).
3. Select the required type and click OK.

Directly from the editor or PHP Explorer:

1. Select a type in the editor or PHP Explorer.
2. Press F4.

Note:

The action will not be activated if the selection is not a resolvable element (i.e. if the selection is not a class name, interface name or class method, constant or field).

The type will be displayed in the Type Hierarchy view.

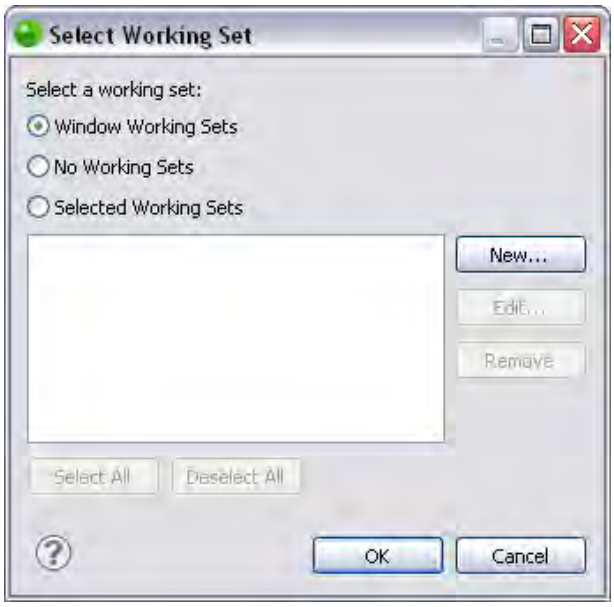
Creating PHP Working Sets

PHP Working Sets can be created from a variety of locations where working sets can be selected.

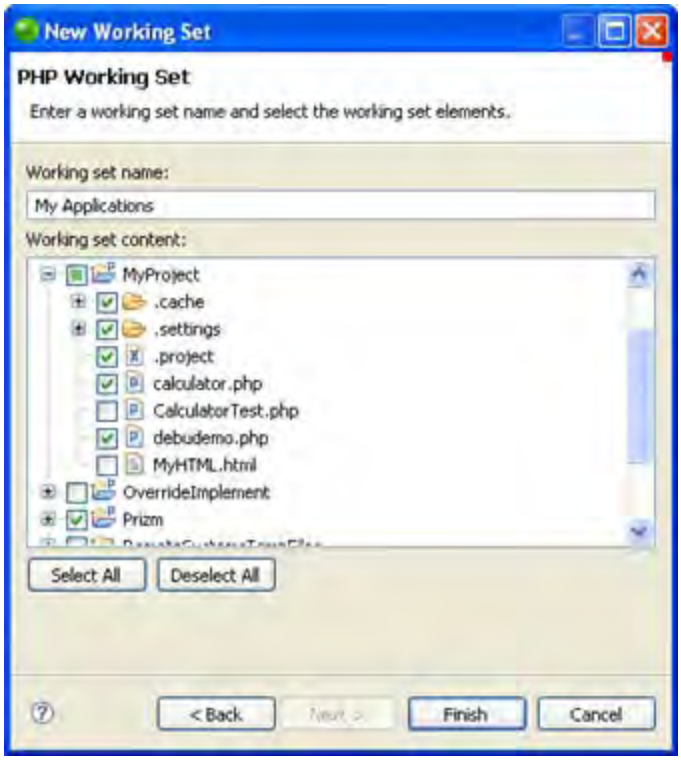
This procedure describes how to create PHP Working Sets from the Window Menu in the PHP Perspective.

To create a PHP Working Set:

1. In the PHP Explorer view click the View Menu arrow .
   The PHP Explorer menu options will open. Select 'Select Working Set...'.
   The Select Working Sets dialog will open.



2. Click New.
   The New Working Set dialog will open.
3. Select PHP and click Next.
   The New PHP Working Set dialog will appear.
4. Enter a name for the Working Set.
5. Mark the checkbox next to the projects/files/folders to be included in the Working Set.



6. Click Finish.

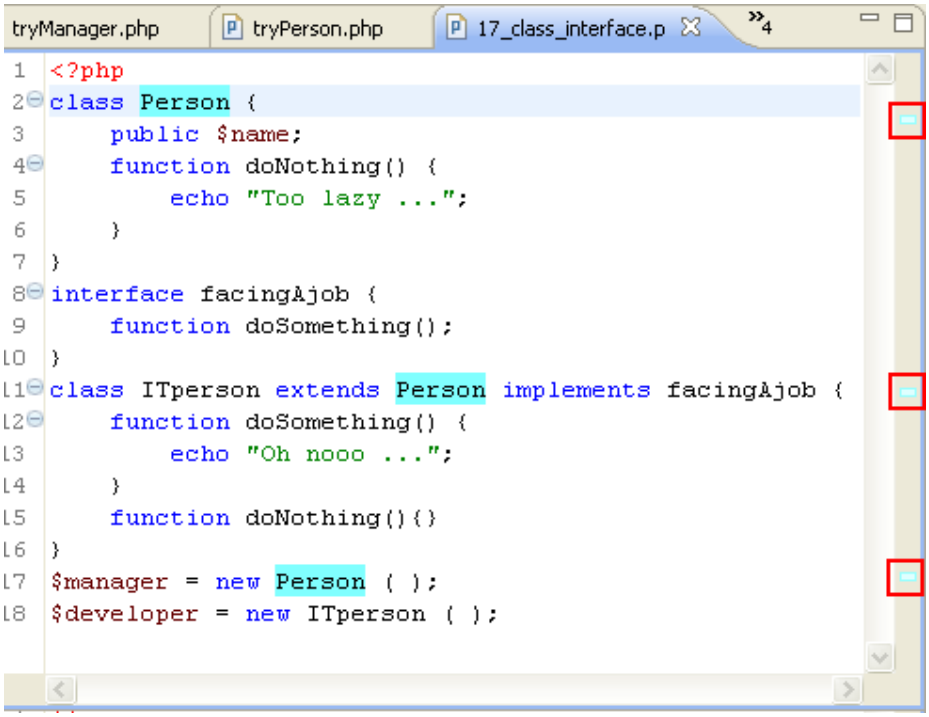
Your new PHP Working Set will be added to the Select Working Sets dialog and will be available for selection for Working Sets actions.

Using Mark Occurrences

The Mark Occurrences feature allows you to see where a variable, method or type is referenced within the active PHP file.

To use Mark Occurrences:

1. Enable the Mark Occurrences feature by clicking the Mark Occurrences icon on the toolbar -or- pressing Alt+Shift+O.
   When the Mark Occurrences feature is enabled, the icon will be highlighted.
2. Place your cursor on the required element in your file.



All instances where the element is referenced within the file will be highlighted and annotations will be displayed in the annotation bar.

Note:

See Mark Occurrences Preferences for information on configuring your Mark Occurrences settings.

Finding and Replacing

This procedure describes how to do a Find and Replace for a string within a file.

To find and replace a string:

1. From within the file, press Crtl+F or from the Menu Bar go to Edit | Find/Replace.
2. A find and replace dialog will appear.

3. Enter the required string to find.
4. Enter the required string to replace it.
5. If necessary, configure the settings under the direction, scope and options category.
6. Click Replace.

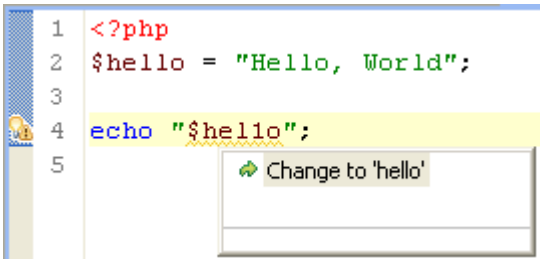
The found string will be replaced by the new string.

Applying Quick Fixes

Quick Fixes are suggestions for corrections to badly written code provided by the Semantic Analysis mechanism which can be easily applied to code to fix code errors.

To apply a quick fix suggestion:

1. Open the file containing the problematic code.
   The line(s) containing code to which a quick fix can be applied will be indicated by a lightbulb icon in the vertical marker bar to the left of the editor window.
   See Real Time Error Detection for more information on how errors are indicated.
2. Place the cursor on the problematic code section and press Ctrl + 1 -or- click the lightbulb icon in the marker bar.
   The Quick Fix list suggests options for fixing the code.



3. Select the required fix from the list by clicking on it or by using the arrow keys and pressing Enter.
4. The selected fix is applied and the code is changed accordingly.

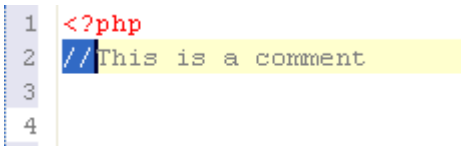
Adding Comments

Zend Studio allows you to quickly and easily comment and uncomment code by selecting a line or a block of text and tagging it as a comment.
Comments can be added to single lines of code (Ctrl + /) or blocks of code (Ctrl + Shift + /).
In addition, special PHPDocBlock comments can also be added. See "Adding PHP DocBlock Comments" for more information.
The following procedures describe how to comment and uncomment lines and blocks of code.

To comment a line:

1. Place the cursor anywhere on the required line of code.
2. Press Ctrl + /

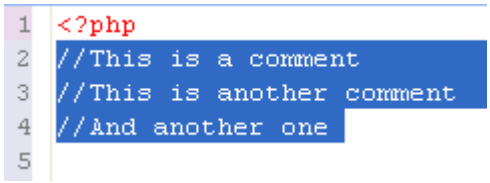
   Two slashes "//" will be added to the front of the line, causing it to be recognized as a comment.



To comment more than one line:

1. Select all the lines that you would like to be commented.
2. Press Ctrl + /

   Two slashes "//" will be added to the front of each line, causing them to be recognized as a comment.



To uncomment a line / lines:

1. Select the required line(s).
2. Press Ctrl + /

   The commenting formatting will be removed from the code.

To comment a block:

1. Select the required block of code.
2. Press Ctrl + Shift + /

   The beginning (/*) and ending (*/) characters will be added in the appropriate places in order to mark the selected block as a comment.

```
1 <?php
2 /*This
3 is a
4 block comment*/
5
```

Adding PHP DocBlock Comments

This procedure describes how to add PHPDoc Comments to PHP elements.

To create a PHPDoc Comment:

In the line above the code for the PHP element, enter the DocBlock characters /** and press Enter.

-Or- Right-click the relevant element in the Outline View and select Source | Add PHPDoc.

A PHPDoc Comment will be created with several parameters to be edited with the relevant information.

The default code comments generated for different elements can be configured through the Code Templates Preferences page.

Using Local History

The following actions can be carried out using the Local History functionality:

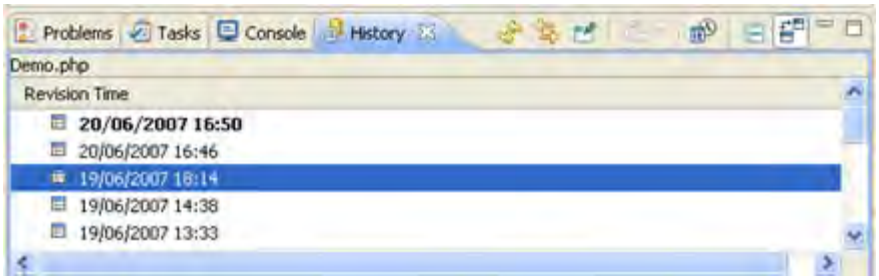
- Comparing Files
- Replacing Files
- Restoring Deleted Files

Comparing Files

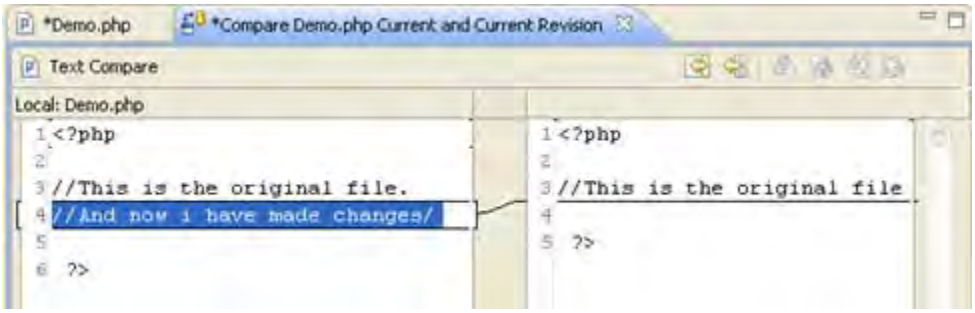
This procedure describes how to compare a current version of a file with one from the Local History.

To compare the current file state and a previous one:

1. Right-click the file in PHP Explorer view and select Compare with | Local history.
   The History View will be displayed with a list of all the previous saved versions of the file.



2. Double-click the version that you would like to view.
   The Text Compare dialog will be displayed, with the current file version displayed in the left pane and the previous version displayed in the right pane.
   Any changes that have been made between the current version and the previous version will be highlighted.



3. Click the next / previous difference buttons to scroll between the differences.

Replacing Files

This procedure describes how to replace a current file with a previous version from the Local History.

To revert a file to one of it's previous states:

1. Right-click the file in PHP Explorer view and select Replace With | Local History.
2. The Compare dialog will be displayed with a list of the previous versions of the file, according to when they were last saved.
3. Double-click a previous version to view it in the Text Compare pane.
4. Once you have found the required version, click Replace.

The current working file will be replaced by the chosen local history file.

Note:

To replace a file with the last saved file, right-click anywhere in the editor and select Replace With | Previous From Local History.

Restoring Deleted Files

This procedure describes how to restore a file that has been deleted.

To restore a deleted file:

1. In PHP Explorer view, right-click the project which previously contained the file and select Restore From Local History.
   A Restore From Local History dialog will be displayed, containing a list of all files that have been deleted from the folder.
2. Select the required file to view its revisions.
3. Select the required revision and click Restore.

The file will be restored into the selected folder.

Using CVS

CVS is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions. The following tasks describe some of the actions that can be done using Zend Studio's CVS functionality:

- Configuring a CVS connection
- Importing a project from CVS
- Accessing an Existing CVS Checkout
- Uploading projects to CVS

See 'Working in a Team Environment with CVS' in the Workbench User Guide for more information on CVS.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Configuring a CVS Connection

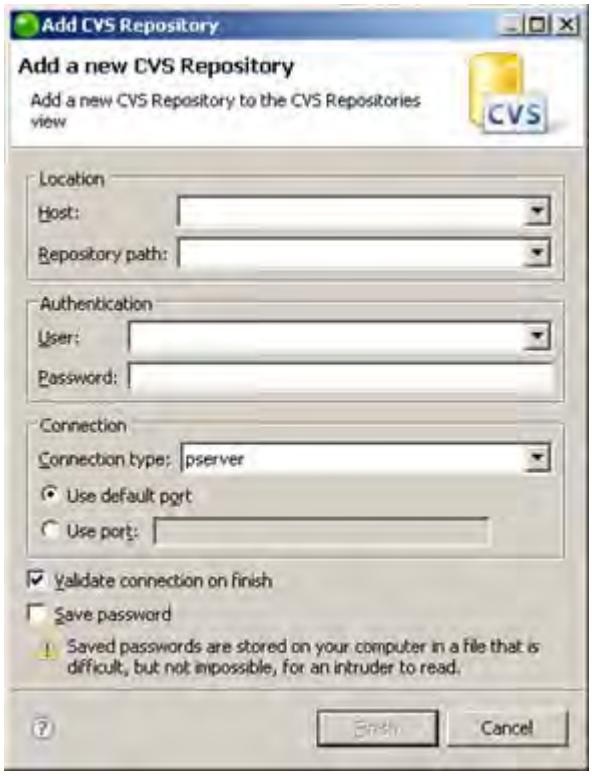
This procedure describes how to configure a connection to a CVS repository:

Before you can add projects to or export projects from CVS, you must define your CVS repository settings.

Prerequisite: To access a repository, make sure that a CVS server is already configured.

To add a new CVS repository:

1. Open the CVS perspective by going to Window | Open Perspective | Other | CVS Repository Exploring.
2. In the CVS Repositories view, click the Add CVS Repository button on the view's toolbar -or- right-click within the view and select New | Repository Location. The Add CVS Repository dialog will open.



3. Enter the information required to identify and connect to the repository location:
   - Host - The host address (e.g. mycomputer.com).
   - Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
   - User - The user name with which you connect to the repository.
   - Password - The password for the user name.
   - Connection Type - The authentication protocol for the CVS server.

     There are four connection methods:

i. pserver - a CVS specific connection method.
ii. extssh - an SSH 2.0 client.
iii. pserverssh2 - provides a pserver connection over ssh2.
iv. ext - the CVS ext connection method which uses an external tool such as SSH to connect to the repository.

- If the host uses a custom port, enable Use Port and enter the port number.

4. Click Finish to create your connection.

Your CVS repository will now be added to the CVS Repository view.

See 'Creating a CVS Repository Location' in the Workbench User Guide for more information.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Importing Projects from CVS

Once projects are placed on the CVS repository, they can be checked out (imported) by anyone with access to that repository. CVS repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.
This procedure describes how to import (check out) projects from an CVS repository location to your desktop.

Note:
If you have resources on your file system which are already linked to CVS, you can access them in Zend Studio by following the instructions under Accessing an Existing CVS Checkout.

To import a project from an CVS repository:

1. Go to File Import | CVS | Projects from CVS.
2. Click Next.
3. Select your repository.
   If you have not yet created a repository, enter the information required to identify and connect to the repository location:
   - Host - The host address (e.g. mycomputer.com).
   - Repository path - The path to the repository on the host (e.g /usr/local/cvsroot)
   - User - The user name with which you connect to the repository.
   - Password - The password for the user name.
   - Connection Type - The authentication protocol for the CVS server.
   - If the host uses a custom port, enable Use Port and enter the port number.
4. Click Next.
5. A 'Select Resource' dialog will appear. Expand the nodes until you see the required project.
6. Select your project and click Finish.
   A 'Check Out As' dialog will appear.
7. Select one of the following options:
   - Check out as project configured using the New Project Wizard - Imports the project as a new PHP project into your workbench - with the project's existing name.
   - Find projects in the children of selected resource - Imports all folders within the project as separate projects.
   - Check out as folder into existing project - Imports the project as a folder into an existing project in your workbench.

- Check out as project with the name specified - Imports the project as a new project into your workbench with a new name. Specify the new name in the box.

Note:
To enable all Zend Studio's PHP functionality for the imported projects, select the 'Check out as a project configured using the New Project Wizard' option and ensure you create the new project as a PHP project.

8. Click Finish.

The project will now be imported into your workspace.

Note that the project will have an CVS repository icon in your PHP explorer view.
Once you have imported a project from CVS into your workspace, you can now add files, edit existing files and commit your changes to the CVS repository.

Note:
Projects can also be checked out from CVS through the CVS Repository Exploring perspective, accessible from Open Perspective | Other | CVS Repository Exploring. Simply right-click the project in CVS Repositories view and select Check Out or Find/Check Out As..

See 'Checking out a project from a CVS repository' in the Workbench User Guide for more information.

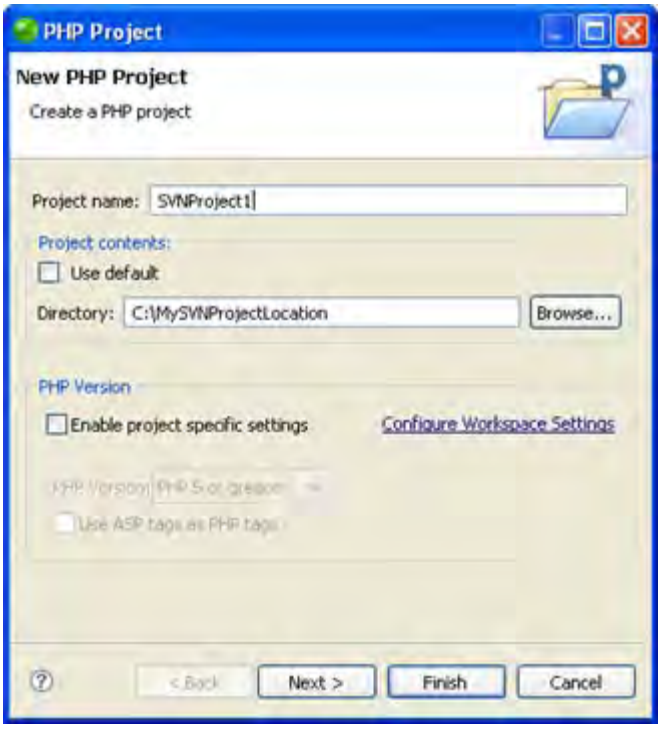
Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Accessing an Existing CVS Checkout

This procedure shows you how to access projects in Zend Studio that you have previously checked out from CVS. This means you do not have to check out resources again to a new location in order to have access to Zend Studio's functionality.

To access previously checked out projects:

1. Launch the New PHP Project Wizard by going to **File | New | PHP Project** -or- right-clicking in PHP Explorer view and selecting **New | PHP Project**.
   The New PHP Project wizard will open.
2. In the Project contents category, unmark the Use default checkbox and browse to the location of your checked out CVS resources.



3. Click Finish.
   A new PHP Project will be created with the contents of the project you had checked out.
4. In PHP Explorer View, right-click your project and select Team | Share Project.
   A Share Project dialog will open.
5. From the repository list, select CVS and click Next.
6. If you have not yet configured a CVS repository, you will need to enter your CVS repository details. See Configuring a CVS Connection for more information.
   If you have already configured a CVS connection, mark the 'Use existing repository

location' option and select your repository from the list.

7. Click Finish.
8. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.

   Enter your password and click Next. (Mark the Save Password checkbox to ensure that this screen does not reappear.)

   A Commit dialog will open.
9. Enter a comment if required and click OK.

Your project will be uploaded to the CVS repository.

Your project will have a repository icon next to it in PHP Explorer view, indicating that it is linked to an CVS repository.

Your can now perform all CVS functions (commit, update etc.) on this project.

Uploading Projects to CVS

Using CVS, you can upload projects and files which other team members can work on.

This procedure describes how to upload a project to your CVS repository location.

Prerequisites: You should have an CVS repository configured before you follow this procedure. See Configuring a CVS Connection for instructions.

To upload a project to an CVS repository:

1. In PHP Explorer View, right-click your project and select Team | Share Project.
   A Share Project dialog will open.
2. From the repository list, select CVS and click Next.
3. Select 'Use existing repository location', and select your repository from the list.
   If you have not yet configured a CVS repository, enter the information required to connect to your CVS Repository. See Configuring a CVS Connection for more information.
4. Click Finish.
5. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click Next. (Mark the Save Password checkbox to ensure that this screen does not reappear.)
   A Commit dialog will open.
6. Enter a comment if required and click OK.

Your project will be uploaded to the CVS repository.

Your project will now have a repository icon next to it in PHP Explorer View, indicating that it is linked to an CVS repository.

Once the project has been committed other team members will be able to access and edit it.

Note:

See 'Sharing a new project using CVS' in the Workbench User Guide for more information.

Using SVN

SVN, or Subversion, is a source control system intended to allow a team or group to work on the same files and projects simultaneously, and to be able to revert file and project states back to previous versions.

The following tasks describe some of the actions that can be done using Zend Studio's SVN functionality:

- Configuring an SVN connection
- Importing a project from SVN
- Accessing an Existing SVN Checkout
- Uploading projects to SVN

See the Subversive User Guide for more information on SVN.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Configuring an SVN Connection

This procedure describes how to configure a connection to an SVN repository.

Before you can add projects to or export projects from SVN, you must define your SVN repository settings.

Prerequisite: To access a repository, make sure that an SVN server is already configured.

To add a new SVN repository:

1. Open the SVN perspective by going to Window | Open Perspective | Other | SVN Repository Exploring.
2. In the SVN Repositories view, click the Add SVN Repository button on the view's toolbar -or- right-click within the SVN view and select New | Repository Location. The Add SVN Repository dialog will open.

3. Enter the information required to identify and connect to the repository location:
   - URL - The URL on which your repository is located.
   - Label - Select whether to use the URL as the repository's name or to enter a new name.
   - Authentication - The user name and password you use to connect to SVN. Mark the Save password checkbox so that the password will be automatically inserted in the future.
4. Click Finish.

Your SVN repository will now be added to the SVN Repository view.

See the Subversive User Guide for more information on SVN.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Importing Projects from SVN

Once projects are placed on the SVN repository, they can be checked out (imported) by anyone with access to that repository. SVN repository connections allow you to import projects from your repository to your workspace, which you can make and upload changes to.

This procedure describes how to import (check out) projects from an SVN repository location to your desktop.

Note:

If you have resources on your file system which are already linked to SVN, you can access them in Zend Studio by following the instructions under Accessing an Existing SVN Checkout.

To import a project from an SVN repository:

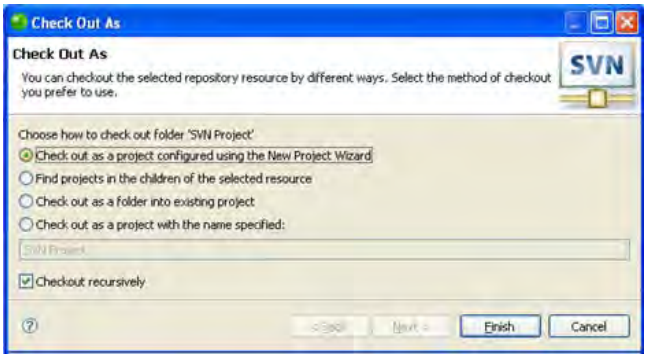
1. Go to File Import | SVN | Projects from SVN.
2. Click Next.
3. Select your repository.

   If you have not yet created a repository, enter the information required to identify and connect to the repository location:

   - URL - The URL on which your repository is located.
   - Label - Select whether to use the URL as the repository's name or to enter a new name.
   - Authentication - The user name and password you use to connect to SVN.

     Mark the Save password checkbox so that the password will be automatically inserted in the future.)
4. Click Next.
5. A 'Select Resource' dialog will appear. Expand the nodes until you see the required project.
6. Select your project and click Finish.

   A 'Check Out As' dialog will appear.


Check Out As
Check Out As
You can checkout the selected repository resource by different ways. Select the method of checkout you prefer to use.
SVN
Choose how to check out folder 'SVN Project'
Check out as a project configured using the New Project Wizard
Find projects in the children of the selected resource
Check out as a folder into existing project
Check out as a project with the name specified:
Checkout recursively
Finish
Cancel

7. Select one of the following options:
   - Check out as project configured using the New Project Wizard - Imports the project as a new PHP project into your workbench - with the project's existing name.
   - Find projects in the children of selected resource - Imports all folders within the project as separate projects.
   - Check out as folder into existing project - Imports the project as a folder into an existing project in your workbench.
   - Check out as project with the name specified - Imports the project as a new project into your workbench with a new name. Specify the new name in the box.

Note:
To enable all Zend Studio's PHP functionality for the imported projects, select the 'Check out as a project configured using the New Project Wizard' option and ensure you create the new project as a PHP project.

8. Click Finish.

The project will now be imported into your workspace.

Note that the project will have an SVN repository icon in your PHP explorer view.
Once you have imported a project from SVN into your workspace, you can now add files, edit existing files and commit your changes to the SVN repository.

Note:
Projects can also be checked out from SVN through the SVN Repository Exploring perspective, accessible from Open Perspective | Other | SVN Repository Exploring. Simply right-click the project in SVN Repositories view and select Check Out or Find/Check Out As..

See the Subversive User Guide for more information on SVN.

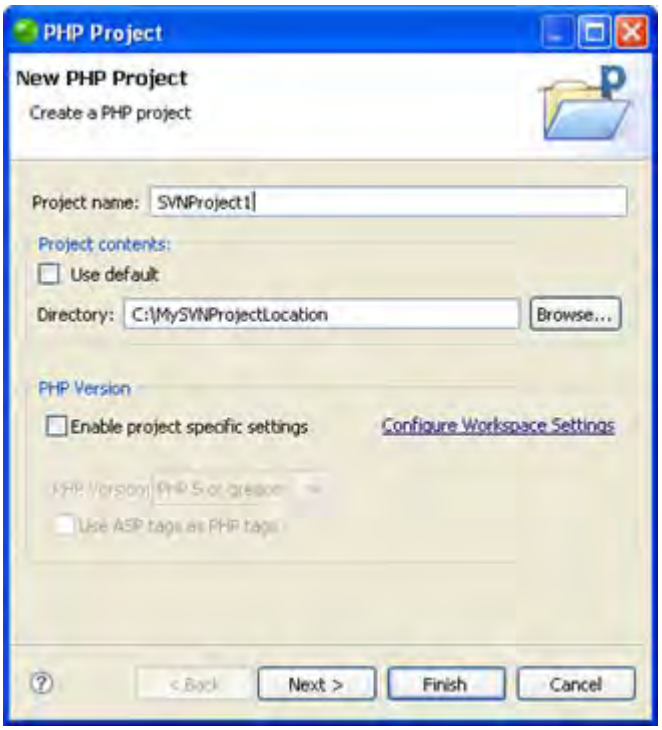
Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Accessing an Existing SVN Checkout

This procedure shows you how to access projects in Zend Studio that you have previously checked out from SVN. This means you do not have to check out resources again to a new location in order to have access to Zend Studio's functionality.

To access previously checked out projects:

1. Launch the New PHP Project Wizard by going to File | New | PHP Project -or- right-clicking in PHP Explorer view and selecting New | PHP Project.
   The New PHP Project wizard will open.
2. In the Project contents category, unmark the Use default checkbox and browse to the location of your checked out SVN resources.



3. Click Finish.
   A new PHP Project will be created with the contents of the project you had checked out.
4. In PHP Explorer View, right-click your project and select Team | Share Project.
   A Share Project dialog will open.
5. From the repository list, select SVN and click Next.
6. If you have not yet configured an SVN repository, you will need to enter your SVN

repository details. See Configuring an SVN Connection for more information.
If you have already configured an SVN connection, mark the 'Use existing repository location' option and select your repository from the list.

7. Click Finish.
8. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Enter your password and click Next. (Mark the Save Password checkbox to ensure that this screen does not reappear.)
   A Commit dialog will open.
9. Enter a comment if required and click OK.

Your project will be uploaded to the SVN repository.

Your project will have a repository icon next to it in PHP Explorer view, indicating that it is linked to an SVN repository.

Your can now perform all SVN functions (commit, update etc.) on this project.

See the Subversive User Guide for more information on SVN.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Uploading Projects to SVN

Using SVN, you can upload projects and files for other team members to access and edit.

This procedure describes how to upload a project to your SVN repository location.

Prerequisites: You should have an SVN repository configured before you follow this procedure. See Configuring an SVN Connection for instructions.

To upload a project to an SVN repository:

1. In PHP Explorer View, right-click your project and select Team | Share Project.
   A Share Project dialog will open.
2. From the repository list, select SVN and click Next.
3. If you have not yet configured an SVN repository, you will need to enter your SVN repository details. See Configuring an SVN Connection for more information.
   If you have already configured an SVN connection, mark the 'Use existing repository location' option and select your repository from the list.
4. Click Finish.
5. Depending on your authentication settings, a dialog might appear asking you to provide authentication information.
   Re-enter your password and click Next. (Mark the Save Password checkbox to ensure that this screen does not reappear.)
   A Commit dialog will open.
6. Enter a comment if required and click OK.

Your project will be uploaded to the SVN repository.

Your project will now have a repository icon next to it in PHP Explorer View, indicating that it is linked to an SVN repository.

Once the project has been committed other team members will be able to access and edit it.

See the Subversive User Guide for more information on SVN.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Using FTP and SSH

Zend Studio's FTP and SSH support is given through the RSE (Remote Server Explorer) plugin. Files from FTP/SSH servers opened in Zend Studio can be edited using Zend Studio's PHP editing functionality such as Debugging, Profiling, Content Assist, Outline views, Formatting and Getter and Setter generation.

To view and edit a file from an FTP/SSH server in Zend Studio, you must first create an FTP/SSH connection.

There are then two ways of working with files linked to FTP/SSH:

1. Viewing and editing files that are located on a remote FTP/SSH server only.
2. Viewing and editing files in your workspace with a link to the FTP/SSH server.

See the RSE User Guide for more on FTP/SSH connectivity.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

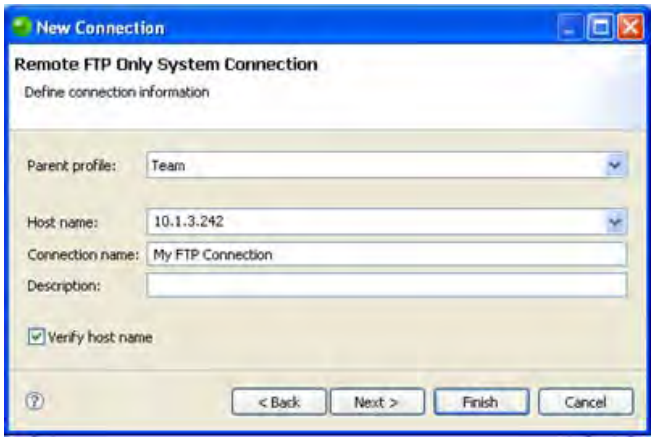
Creating an FTP/SSH Connection

This procedure describes how you can configure an FTP or SFTP connection in Zend Studio.

To create an FTP/SSH Connection:

1. Open the Remote Systems Perspective by going to **Window | Open Perspective | Other | Remote Systems Explorer**.
2. Open the Remote Systems view by going to **Window | Show View | Other | Remote Systems | Remote Systems | Remote Systems**.
3. Click the 'Define a connection to remote system' button on the view's toolbar.
4. The New Connection dialog will appear.

5. Select 'FTP Only' for an FTP connection or 'SSH Only' for an SSH connection and click Next.



6. Enter the following Remote FTP/SSH connection details:
   - Parent Profile - Select a Profile from the drop-down list.
   - Host Name - Enter a valid Host name or IP address for the FTP/SSH Server.
   - Connection name - Enter a Connection Name. A Connection Name is an alias that identifies the server in the Zend Studio file system.
   - Description - Enter a description to help identify your connection (optional.)
   - Verify host name - Mark the checkbox for Zend Studio to verify the existence of the host name before the connection is created.
7. Click Finish.

Your connection will be created and listed in the Remote Systems view.

For information on editing the default port of your FTP/SSH connection see Editing the Default Port of an FTP/SSH Connection.

Editing the Default Port of an FTP/SSH Connection

This procedure describes how to edit the default port of an FTP/SSH connection. In order to edit the default port you must have an FTP/SSH connection set up. For more information see Creating and FTP/SSH Connection.

To edit the default port of an FTP/SSH connection:

1. Open the Remote Systems view by going to **Window | Show View | Other | Remote Systems | Remote Systems | Remote Systems**.
2. Select the "Files" sub-folder of the connection.
3. From the right click menu select **Properties | Subsystem.**
   The "Subsystem" dialog opens.
4. In the "Port" text field edit the port number.
   If the "Port" text field does not allow you to edit upon opening, click to use the first available port or explicitly set the port number.
5. Click **OK** to save all changes.
   The port number for your FTP/SSH connection has been edited.

To change the FTP/SSH connection port number back to the default port (or the first available port) click .

See the RSE User Guide for more on FTP/SSH connectivity.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

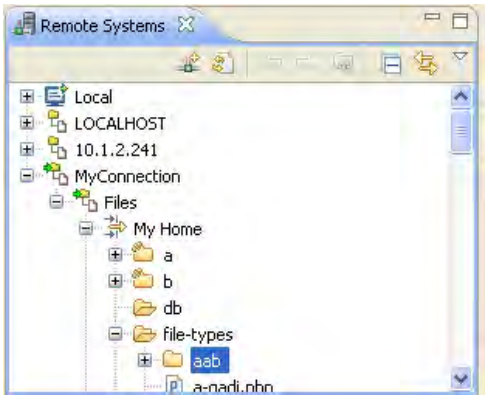
Viewing and Editing Files on an FTP/SSH Server

Files located on an FTP/SSH server can be opened and edited in Zend Studio without having to save them to your Workspace.

This procedure describes how to view and edit files which are situated on your FTP/SFTP server.

To view and edit files located on an FTP/SSH server:

1. Create an FTP/SSH connection by following the steps under the "Creating an FTP/SSH Connection" topic.
2. Open the Remote Systems view by going to Window | Show View | Remote Systems | Remote Systems.
3. Expand the tree under your FTP/SSH connection.



4. Some files and folders might require you to enter your FTP/SSH login User ID and Password:

- Double-click a PHP file to open it in a Zend Studio PHP editor.

  Most of Zend Studio's functionality, such as Content Assist, Formatting, Outline Models and Getter and Setter Generation will be available to the file.

5. Edit the file as required.
6. Click Save.

The edits will be saved on the FTP version of the file.

Viewing and Editing Remote Server Files in your Workspace

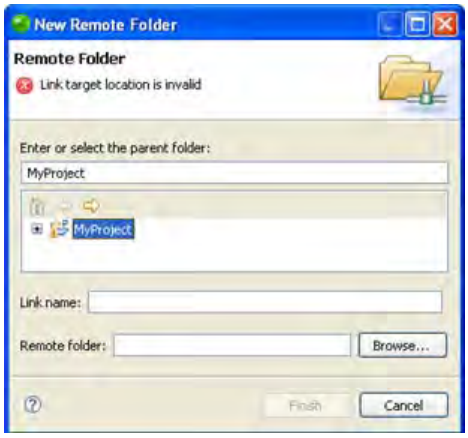
Folders situated on FTP/SSH servers can be saved onto your Workspace while maintaining the link with the FTP/SSH server.

Any changes made in the files and folders located on either the FTP/SSH servers or the Workspace will be updated in both of the file locations.

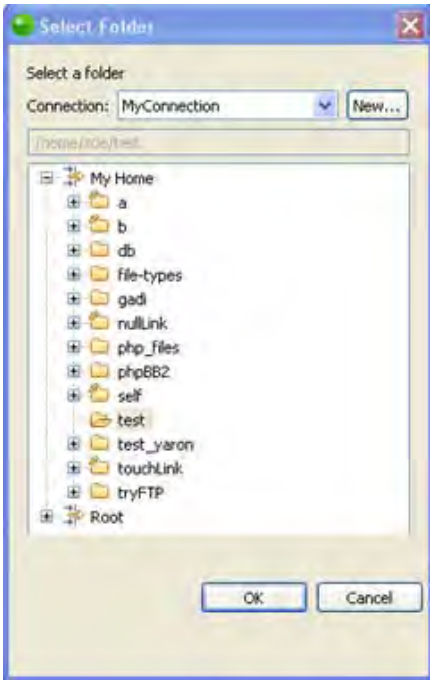
This procedure describes how to simultaneously edit files and folders on your FTP server and your Workspace.

To save, view and edit files in your Workspace which are located on an FTP/SSH server:

1. In PHP Explorer view, right-click and select New | Remote Folder -or- go to File on the Menu Bar and select New | Remote Folder.
2. The New Remote Folder creation wizard will open.



3. Select the project in which you want to place your FTP/SSH folder.
4. In the Remote folder selection, click Browse.
   The Select Folder dialog will open.



5. In the 'Connection' drop-down list, select your FTP/SSH connection.
   If you have not yet established a connection, click New and follow the instructions under Creating an FTP/SSH Connection.
6. Your FTP/SSH file system will be displayed.
7. Select the folder you would like to link to your Workspace and click OK.
8. You will be returned to the New Remote Folder dialog.

9. The link name will be automatically be updated with the name of your folder. Edit this if required.

10. Click Finish.

Your FTP/SSH folder and its contents will be added to your Workspace.

Most of Zend Studio's functionality, such as Content Assist, Formatting, Outline Models and Getter and Setter Generation will be available to help you edit the file.

Any changes made to the files and folders in your Workspace will be updated in the FTP copy, and conversely any changes made on the FTP copies will be updated in your Workspace.

Folders that are linked to FTP will be displayed with a linked connection icon .

Note:

Deleting an FTP/SSH folder from your Workspace will only delete it from your Workspace. However, deleting a file contained within a folder will also delete the file on the FTP/SSH server. Adding new files to the linked folder in your workspace will upload the new files to your FTP/SSH server.

Developing with Zend Framework

When developing using the Zend Framework, you should first create a Zend Framework project, which has the Zend Framework library added to it's include path.

You should then use the New Zend Framework item wizard to add Zend Framework elements to your project.

- For information on how to create Zend Framework Projects, see Creating Zend Framework Projects.
- To trial using Zend Framework, you can create a Zend Framework Example Project.
- For information on creating Zend Modules, Zend Controllers, Zend Tables, Zend Views, Zend View Helpers and Zend Action Helpers , see Creating Zend Framework Elements.
- For information on how to search for information on Zend Framework, see Searching the Zend Framework Site

Once you have created your Zend Framework application, you can run or debug it on your server by following the instructions in Running and Debugging Zend Framework Projects.

Creating Zend Framework Projects

Zend Framework Projects are PHP projects which contain Zend Framework's libraries in their Include Path and which are organized into Framework's Controller-Model- View format (if selected). Zend Framework 1.8 projects are based on the Zend Tool, ensuring your projects comply with the latest Zend Framework standards and allow for Rapid Application Development . Visit http://framework.zend.com for more on Zend Framework or http://framework.zend.com/manual/en for the Zend Framework Reference manual.

This procedure demonstrates how to create a new Zend Framework Project.

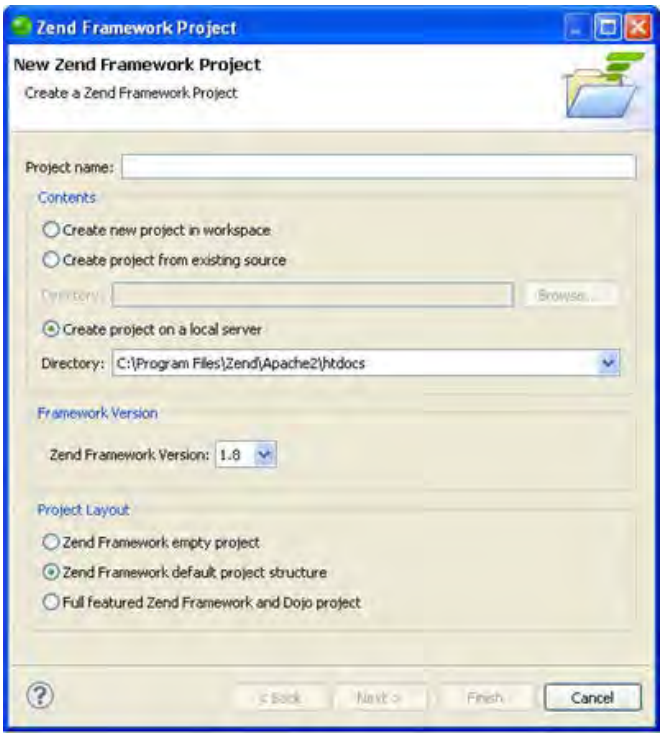
Important Note:

You must have Zend Framework version 1.8 installed on your machine to be able to create Zend Framework projects.

Zend Framework comes bundled with Zend Server or can be downloaded from the Zend Framework site at http://framework.zend.com/download/current.

To create a new Zend Framework Project:

1. Go to File | New | Zend Framework Project -or- right-click in PHP Explorer view and select File | New | Zend Framework Project.
   The New Zend Framework Project dialog will open.



2. Enter the project name.
3. In the Framework Version drop-down list, select the Zend Framework version to use for your project. Zend Studio supports Zend Framework version**1.9**. This will include the required Zend Framework library in your project, and give you access to the relevant content assist options.
4. Select the Project Layout from the following options:
   - Zend Framework empty project - Creates a regular PHP project structure
   - Zend Framework default project structure - Creates a project organized according to Zend Tool conventions.
   - Full featured Zend Framework and Dojo project - Select this option if you are going to be developing using the JavaScript Dojo framework in this project. This will create a link to the external Dojo toolkit library. See Setting Up and Using Dojo Integration for more information.
5. Click Finish.

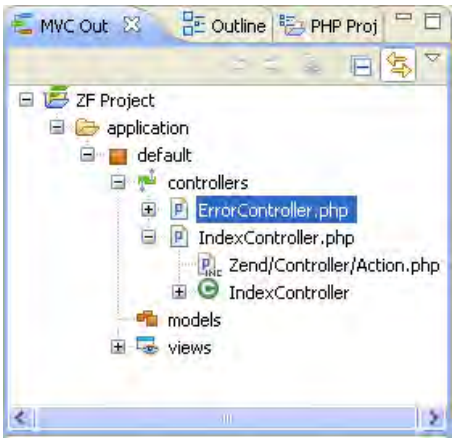
   Your Zend Framework Project will be created.

   It will be represented by a Zend Framework icon in PHP Explorer view.
6. Click Yes when prompted to open the Zend Framework perspective.

   Your Zend Framework project will be displayed in the MVC Outline view.

Note:

To open the MVC Outline view manually, go to Window | Show View | PHP Tools | MVC Outline.



The MVC view provides an outline of all controller, model and view classes, files, variables and related functions.

You can now start to add your own Model, View and Controller files. See Creating Zend Framework Elements for more information.

Running and Debugging Zend Framework Projects

In order to run and debug Zend Framework projects on your server, you will need to set up your server to be able to execute Zend Framework applications.

Setting Up Your Server Environment

Prerequisites:

- Apache 2.2 or later*
- PHP 5*
- Zend Framework*

  Note: The Zend Framework version installed on your server should match the Zend Framework version in which you developed your application.

* These components can be installed together by installing Zend Server.

The following instructions will guide you through the process of setting up your Zend Server in order to be able to run Zend Framework-based applications.

Note:

Zend Server is the recommended environment but Zend Framework projects can be run on other servers. Different configuration information might apply.

To set up your server in order to run/debug Zend Framework Projects:

1. If you do not have a server running a PHP distribution already installed, you can install Zend Server (downloadable from http://www.zend.com/en/products/server). Ensure you install with the following settings:
   i. If you do not have a Web server, install the Apache bundled with Zend Server.
   ii. If you do not have the Zend Framework files already installed on your machine, install Zend Server in 'Complete' mode, or ensure that Zend Framework is selected in the list of additional components in 'Custom' mode.
2. If you choose not to install Zend Server make sure you install the following:
   i. PHP 5
   ii. Apache 2
   iii. Zend Framework files (downloadable from http://framework.zend.com/download).
3. Make sure your php.ini settings are properly configured as follows: (These can be configured through Zend Server or by manually configuring your php.ini file).

Note:
If you are using Zend Server, these settings should be configured by default.

i. Your include path is pointing to your Zend Framework library (e.g. .;C:\Program Files\Zend\ZendServer\share\ZendFramework\library).
   This is configured in the 'include_path' directive.
   In Zend Server, this directive can be found under the 'Paths and Directories' category in Server Setup | Directives.

ii. The session settings are configured to save path points to a temporary directory writable by the Web server (e.g."C:/temp" on Windows or "/tmp" on UNIX / Linux.)
   This is configured in the 'session.save_path' directive.
   In Zend Server, this directive can be found under the 'session' category in Server Setup | Directives.

iii. PHP Short Tags are enabled.
   This is configured in the short_open_tag directive.
   In Zend Server, this directive is on by default. It can be found under the 'Language Options' category in Server Setup | Directives.

iv. If you want to display errors, set your display error setting to on.
   This is configured in the display_errors directive.
   In Zend Server, this can found under the 'Error Handling and Logging' category in Server Setup | Directives.

4. Restart your Web server after applying changes.

Once you copy your Zend Framework project files to the server you will be able to run and debug the project as you would a normal PHP project.

To configure Zend Server to run a Zend Framework application:

1. Define a virtual host on Zend Server that will point to the new project's public directory:

i. Find the full path to your project's public directory.
   This is listed in the location field in the project's properties. To see it, in PHP Explorer view right-click the public directory and select Properties | Resources.

ii. Open your Apache configuration file (in most cases it will be httpd.conf and located in your Apache installation directory).
   See Where is my Apache configuration file for more information.

ii. Go to the end of the file and add the following code:

```
Listen 10089
```

```
<VirtualHost *:10089>
    DocumentRoot "DOCUMENT_ROOT"
    <Directory "DOCUMENT_ROOT">
        Order allow,deny
        Allow from all
    AllowOverride all
</Directory>
</VirtualHost>
```

2. Replace "DOCUMENT_ROOT" with the full path to the public directory, enclosed in double quotes ("DOCUMENT_ROOT").
   Replace the port number with a unique port number dedicated to this Virtual Host. The port number (10089) has to be the same value for "Listen" and "VirtualHost".
3. Zend Framework's MVC implementation makes use of the Front Controller pattern. You must therefore rewrite all incoming requests (except those for static resources, which your application need not handle) to a single script that will initialize the FrontController and route the request. If you're using mod_rewrite for the Apache web server, create the file /public/.htaccess with the following contents:

```
# public/.htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ /index.php [NC,L]
```

Note:

Some web servers may ignore .htaccess files unless otherwise configured. Make sure that your web server is configured to read the .htaccess file in your public directory.

4. Restart your Web server from the command line (windows user can use the Apache Monitor tool).

Once you copy your Zend Framework project files to the server you will be able to run and debug the project as you would a normal PHP project.

Where is My Apache Configuration File?

Apache uses a main configuration file for all its settings, typically this file is called httpd.conf or apache2.conf. The location of this file varies depending on your installation:

- **Windows:**
  - <install_dir>\Apache2.2\conf\httpd.conf
  - If you changed the location of your Zend Server installation, your document root will be located at <installation_directory>\ Apache2.2\conf\httod.conf, where <installation_directory> is the location of the directory in which Zend Server is installed.
- **Linux**:
    - If you installed Zend Server from a repository (DEB or RPM packages), the location of your configuration file is defined by your distribution's Apache packages, and will vary depending on your distribution and configuration.
- Common locations include:
    - Debian / Ubuntu - /etc/apache2/apache2.conf
    - Fedora Core / RHEL / CentOS - /etc/httpd/httpd.conf
- If you installed Zend Server using the generic Tarball package - /usr/local/zend/apache2/conf/httpd.conf.
- If you changed the location of your Zend Server installation, your document root will be located at <installation_directory>/ apache2/conf/httpd.conf, where <installation_directory> is the location of the directory in which Zend Server is installed.

Creating and Running a Zend Framework Example Project

The purpose of the Zend Framework Example Project is to demonstrate the capabilities and best practices of working with the Zend Framework.

Zend Framework is a high quality open source framework for developing Web Applications and Web Services with PHP. The Zend Framework Example Project is a small demo application that shows how Zend Framework can be leveraged in order to streamline writing code, by decreasing development time (through using prewritten modules) and demonstrating how to implement best practices in organizing your project.

Once the Example Project has been created, you will be able to navigate through the tree in PHP Explorer and see the different components of the project.

The Zend Framework Example provides a model called MVC (Model-View-Controller) for programming using the Framework. With this model, applications are divided into three parts to assist in making the development process more efficient.

The components are:

- Controller - includes all code that handles the logic.
- Model - contains data access commands to the raw data.
- View - contains the application's front end (User Interface).

Additional information about programming with the Zend Framework can be found in the official Zend Framework Manual at: http://framework.zend.com/manual

For more information about the MVC method see http://en.wikipedia.org/wiki/Model-view-controller.

Creating the Zend Framework Example Project

The following procedure describes how to create a Zend Framework example project.

To create a Zend Framework Example Project:

1. Go to **File | New | Example | Zend Framework Example Project**.
   The "New Zend Framework Example Project" wizard will open.
2. Enter a name for your example project.
3. You have two options for the contents of your Zend Framework Example Project:
   - Create new project in workspace - Creates the project in your current workspace.
   - Create project at existing location - Creates the project from an existing location outside of your workspace.

   You may choose your location by writing it in the "Directory" text field, or by clicking **Browse...**
4. Click **Finish**
   An example project will be created and displayed in the PHP Explorer view.

You can navigate through the application source files to learn more about the source behind the application.

Running the Zend Framework Example Project

The following procedure describes how to run the Zend Framework example project. In order to run your Zend Framework Example Project, you must first create one. For more information see Creating the Zend Framework Example Project.

Note:

You must have an established connection with a server to run the Zend Framework Example project. For more information see DefiningZend Server in Zend Studio.

To run the Zend Framework example project:

1. Expand your Zend Framework Example project in the PHP Explorer View.
   The project's sub-folders will be exposed.
2. Expand the sub-folder called **public** and right click on the sub-file **index.php.**
   From the right click menu select **Run as... | PHP Web Page.**
   The "Run PHP Web Page" dialog will open.
   If the dialog doesn't open you will need to add this directory to the build path by right clicking on the Public folder and selecting **Build Path | Include**.
   You do not need to make any changes in order to run the example project.
3. Click **OK.**

Your project web page opens.

Creating Zend Framework Elements

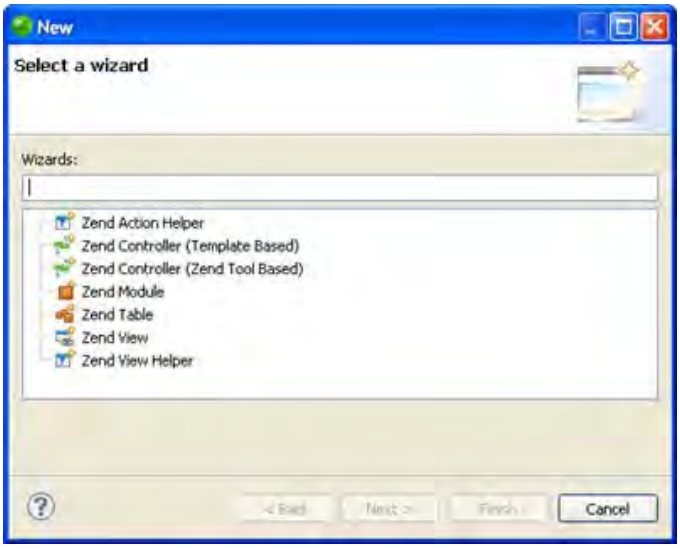
Once you have created a Zend Framework Project, you can add Zend Framework elements through the Zend Framework Item wizards.

Elements should be placed within a 'Zend Module', which enables you to organize your MVC files.

Note:

When working with Zend Framework files, you should open the Zend Framework perspective by going to Window | Open Perspective | Zend Framework.

Zend Framework items can be created through the New Zend Framework Item dialog, accessible by going to New | Zend Framework Item.



Through this dialog, you can create the following elements:

- Zend Action Helper
- Zend Controller
- Zend Module
- Zend Table
- Zend View
- Zend View Helper

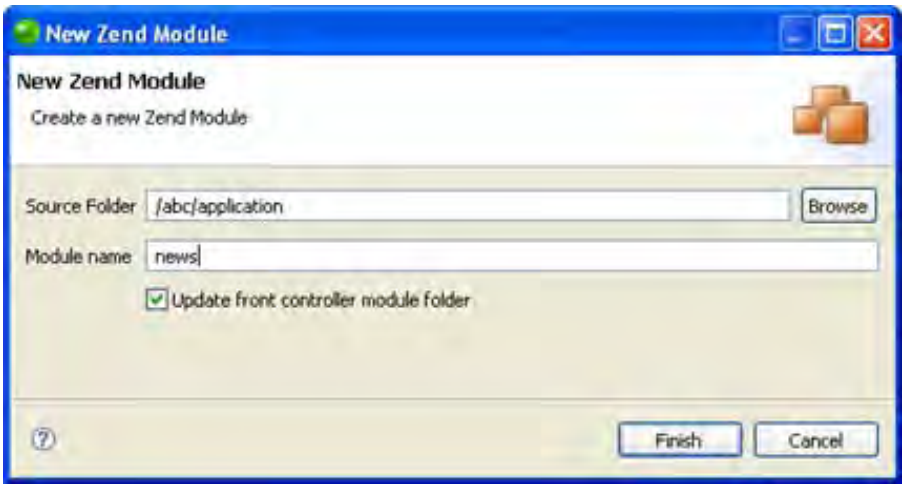
Creating a Zend Module File

Zend Framework MVC files should be created within a Zend Module, which enables you to easily group your MVC files according to your application's components. This will help you organize your Zend Framework Project structure.
One Zend Module, entitled 'default', will be automatically created within the Zend Framework project. Further Zend Modules can be created to group your MVC files according to your needs.

Note:
When working with Zend Framework files, you should open the Zend Framework perspective by going to **Window | Open Perspective | Zend Framework**.

To create a Zend Module:

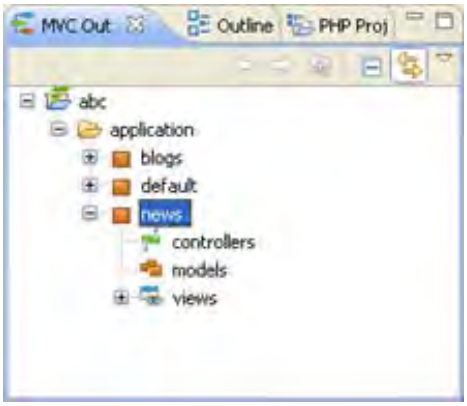
1. In PHP Explorer view, right-click the Zend Framework project in which you want to create the Zend Module and select **New | Other | Zend Framework | Zend Module**. The New Zend Module wizard will open.



2. The folder in which the Zend Module should be created is the project's 'application' folder. If necessary, click Browse to change the Source Folder.
3. In the Module name category, enter a name for the new Zend Module.
4. Leave the 'Update front controller module folder' checkbox marked to add a line in the bootstrap file that specifies the new module.
5. Click Finish.

The new Zend Module will be created in your project and will be displayed in the MVC Outline view, represented by an orange square .

The Zend Module will include controller, model and view folders in the relevant file hierarchy.



Creating a Zend Controller File

This procedure describes how to create a new Zend Controller file.

A Zend Controller is a class for working with the "controller" portion of the model-view-controller pattern. The Zend Controller manages the communication between user actions and your application.
See http://framework.zend.com/manual/en/zend.controller.html for more information on the Zend Controller.

Zend Studio offers the option to create Zend Controller files based on the Zend Controller Template (for projects based on Zend Framework version 1.7 and below) and based on the Zend Tool (for projects based on Zend Framework version 1.8 and above).

To create a new Zend Controller file:

1. In PHP Explorer view, right-click the controllers folder in your Zend Framework Project and select New | Zend Framework Item | Zend Controller.
   The New Zend Controller Wizard is displayed.

2. If you selected to create a Zend Tool based Controller:
   i. Enter a name for the Controller.
      The default name for the created controller will be <Controller Name>Controller.php (e.g. MyNewController)
      The default location for the file will be in the application's default 'controllers' folder.
   ii. Select the required containing Zend Framework project from the Project Name

drop-down list.

If you selected to create a Template-based Controller:

i. Ensure the required containing source folder is correct.
   It is recommended to create all Zend Controllers in the project's controllers folder.
ii. Enter the Controller name in the File Name field.
iii. Click Next to view and edit the template on which the new Controller will be created.

3. Click Finish.

A new Zend Controller file will be created with the relevant template.

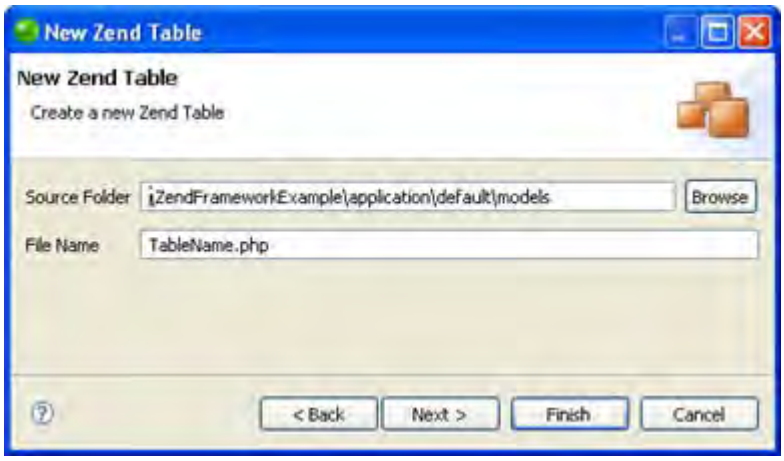
The new file will be created in the project's 'controllers' folders and displayed in the MVC Outline view.

Creating a Zend Table File

This procedure describes how to create a new Zend Table file.
A Zend Table is a class for working with the "model" portion of the model-view-controller pattern.
The Zend Table manages the information and data manipulation in your application.

To create a new Zend Table file:

1. In PHP Explorer view, right-click the models folder in your Zend Framework Project and select New | Zend Framework Item | Zend Table.
   The New Zend Table Wizard will be displayed.



2. The default location for the file will be in the application's 'models' folder.
   Click Browse next to the Source Folder field to change the location.
3. Edit the file name.
4. Click Finish.

A new Zend Table file will be created with the Zend Table template.
The new file will be displayed in the MVC Oultine view.

Creating a Zend View File

This procedure describes how to create a new Zend View file.
A Zend View is a class for working with the "view" portion of the model-view-controller pattern.
The Zend View manages the graphic interface aspect of your application.
See http://framework.zend.com/manual/en/zend.view.html for more information on the Zend View.

To create a new Zend View file:

1. In PHP Explorer view, right-click the views folder in your Zend Framework Project and select New | Zend Framework Item | Zend View.
   The New Zend View Wizard will be displayed.

2. The default location for the file will be in the application's default 'views' folder. Click Browse next to the Source Folder field to change the location.
3. Edit the File Name if required.
4. Click Finish.

A new Zend View file will be created with the relevant template.
The new file will be displayed in the MVC Oultine view.

Creating a Zend View Helper File

This procedure describes how to create a new Zend View Helper File. Zend View Helpers, when attached to a view object, can call the helper as if it were a method of the view object itself. The View object retains helper instances, which means that they retain states between calls.
See http://framework.zend.com/manual/en/zend.view.helpers.html for more on Zend View Helpers.

To create a new Zend View Helper file:

1. In PHP Explorer view, right-click the relevant helpers folder in your Zend Framework Project and select New | Zend Framework Item | Zend View Helper.
   The New Zend View Helper Wizard will be displayed.
2. Ensure the source folder is correct or click Browse to change.
3. Enter the Helper's Prefix.
4. Enter the Helper's name. This will be the name of the Helper file.
   The Helper's class name will be automatically created in the format <Helper's_Prefix>_<Helper's_Name>.
5. Click Next.
6. In the Select PHP Template dialog, ensure the New Zend View Helper template is selected.
7. Click Finish.

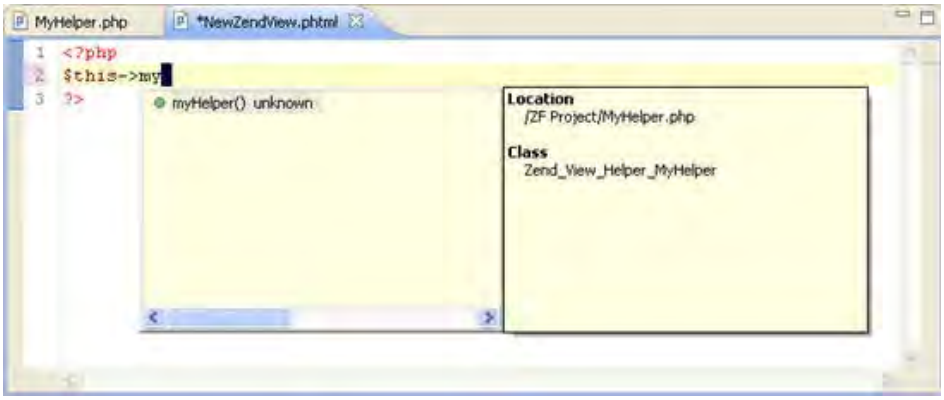
A new Zend View Helper file will be created with the relevant template. This includes phpDoc block comments which help Zend Studio to recognize that the element is a Zend View Helper.

Note:

Zend View Helper phpDoc block comments must be in the format:

```
/**
* <Helper's_Name> helper
*
* @uses viewHelper <Helper's_Prefix>
*/
```

All Zend View Helpers in your project which are correctly commented will be available in the Content Assist list in a Zend View file. This includes the default Zend View Helpers included in Zend Framework's libraries.



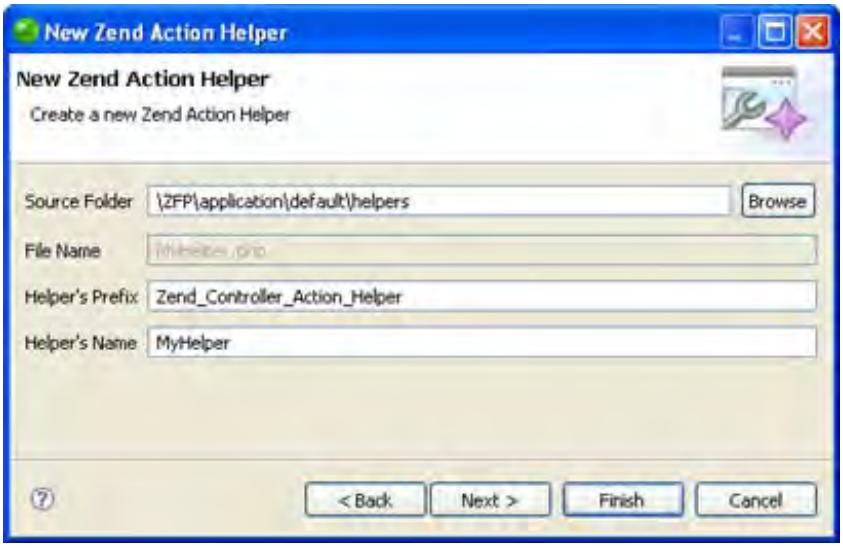
In addition, pressing Ctrl and clicking on a Zend View Helper defined in a View file will take you to the Zend View Helper's declaration.

Creating a Zend Action Helper

Zend Action Helpers provide an easy way of extending the capabilities of Action Controllers, allowing you to extend Action Controller functionality only when it is needed. Zend Studio allows you to easily create and use Action Helpers within your Zend Framework projects.
For more information on Action Helpers, see http://devzone.zend.com/article/3350-Action-Helpers-in-Zend-Framework.

To create a new Zend Action Helper file:

1. In PHP Explorer view, right-click the relevant helpers folder in your Zend Framework Project and select New | Zend Framework Item | Zend Action Helper.
2. The New Zend Action Helper Wizard will be displayed.



3. The default location for the file will be in the application's default 'helper' folder. Click Browse next to the Source Folder field to change the location.
4. Enter the Helper's name. This will be the name of the Helper file.
   The Helper's class name will be automatically created in the format <Helper's_Prefix>_<Helper's_Name>.
5. Click Finish.

A new Zend Action Helper file will be created with the relevant template. This includes phpDoc block comments which help Zend Studio to recognize that the element is a Zend Action Helper.

Note:

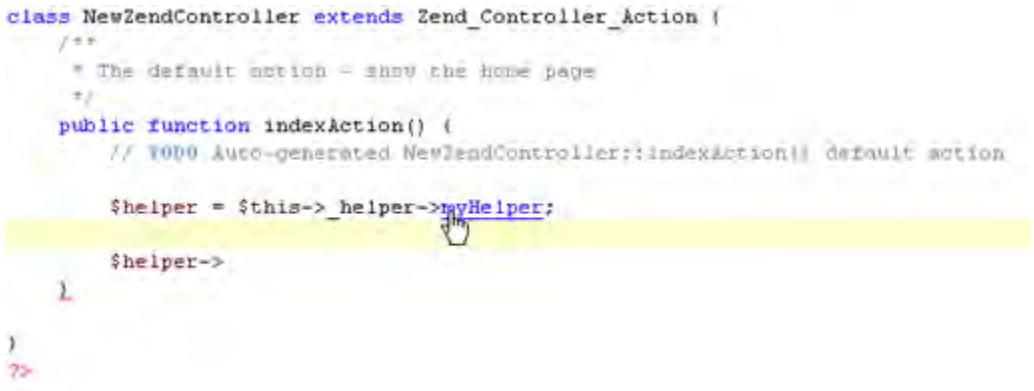
Zend Action Helper phpDoc block comments must be in the format:

```
/**
* <Helper's_Name> Action Helper
*
* @uses actionHelper <Helper's_Prefix>
*/
```

All Zend Action Helpers in your project which are correctly commented will be available in the Code Assist list in Zend Controller files. This includes both the default Zend Action Helpers included in Zend Framework's libraries and the Actions Helpers you created.

The Helper broker for the Helper's members and methods are also available for Content Assist:

In addition, pressing Ctrl and clicking on a Zend Action Helper in a Controller file will take you to the Zend Controller Action Helper's declaration.

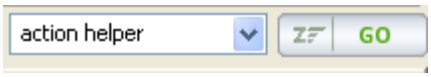


Searching the Zend Framework Site

Zend Studio's workbench contains a search box which allows you to quickly and easily retrieve information from the Zend Framework site, without having to leave the application.
This procedure describes how to search for information in the Zend Framework site.

To search the Zend Framework site:

1. Open the Zend Framework Perspective by going to Window | Open Perspective | Zend Framework.
   The Workbench will switch to the Zend Framework Perspective and the Zend Framework search box will appear in the bottom right of the Workspace.
2. Enter a term in the Zend Framework search box.



3. Click the 'Go' button.
   A new browser window will open in Zend Studio, displaying a list of relevant results from the Zend Framework site.

Note:
The browser will open according to the settings defined in Window | Preferences | Web Browser.

4. Click the required result to be taken to the page containing the relevant information.

Using the Zend Tool Floating Window

Zend Studio provides you with the ability to execute Zend_Tool commands from within Zend Studio using the Zend Tool Floating Window. This replicates entering a command in your CLI (command line interface).

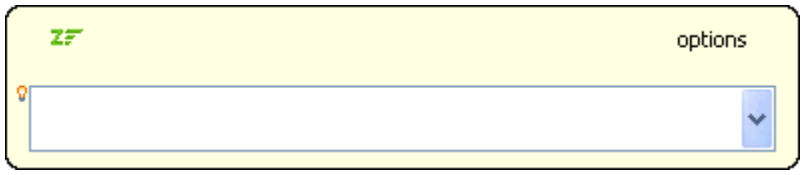
The Zend_Tool Initative allows PHP programmers to programme according to Rapid Application Development principles. See http://framework.zend.com/wiki/display/ZFDEV/Zend+Tool+Initiative for more information on the Zend_Tool initiative.

Note:

You must have Zend Framework version 1.8 installed and configured in your php.ini file for Zend_Tool commands to function.
The latest version of Zend Framework comes bundled with Zend Server or can be downloaded from the Zend Framework site at http://framework.zend.com/download/current.

To execute a Zend_Tool command:

1. In PHP Explorer view, select a Zend Framework project.
2. Press Ctrl + 2 -or- from the menu bar go to Project | Zend Tool.
   The Zend Tool Floating Window displays.

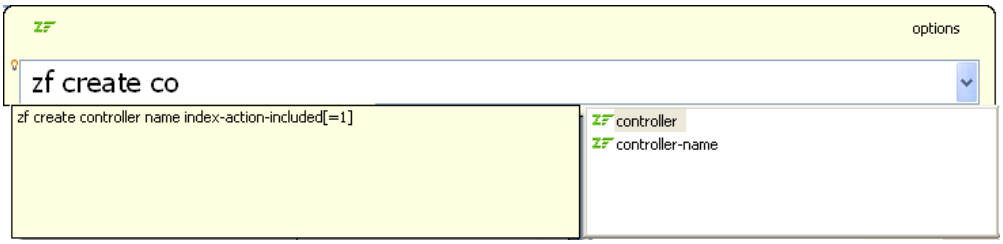


3. Enter the required command.

Note:

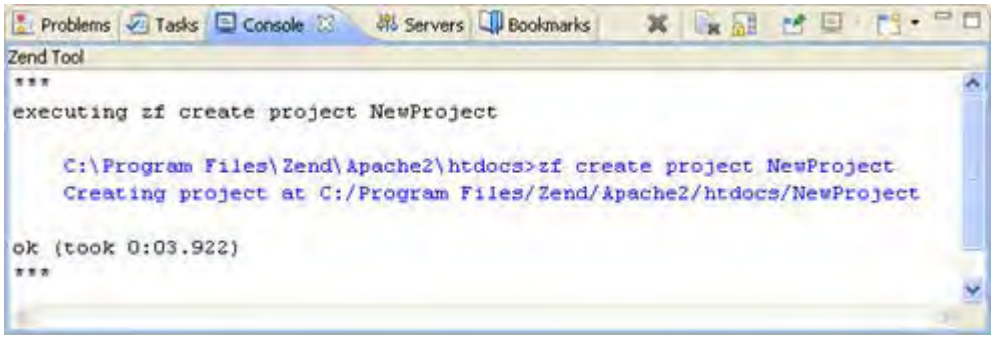
The Zend_Tool currently supports a limited number of functions.

The content assist list displays possible options as you type.



4. Press Enter.

The command will be executed and details will be displayed in the Zend Tool Console view.



See Zend Framework Preferences for information on customizing the output displayed in the Zend Tool Console.

Connecting to Databases

Connecting to Databases

The following tasks describe how to connect to and interact with a database using the Data Tools Platform:

- Creating a Database Connection Profile
- Connecting to a Database
- Viewing and Editing Database Table Content
- Creating and Executing an SQL Query

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Creating a Database Connection Profile

This procedure describes how to create a connection profile to your database, allowing you to easily connect to it through the Database Development perspective.

The drivers for most of the database connection types are automatically included and configured in Zend Studio.

However, some drivers require you to manually add and configure the driver archive files.

The following driver types come bundled and preconfigured in Zend Studio:

- DB2
- SB2 i-series
- Derby
- MySQL
- Oracle
- Postre SQL
- SQL Server

To create a connection profile:

1. In the PHP Perspective toolbar, click the Create New SQL Connection icon
   -Or- in the Database Development perspective (**Window | Open Perspective | Other | Database Development**), click the Create New SQL Connection icon in the Data Source Explorer view or on the main Toolbar
   -Or- right-clickthe Database Connections folder int he Data Source Explorer viee and select New.
2. The New Connection Profile wizard opens.

3. Select the required connection profile type from the list.
4. Enter a name for the connection and a description (if required).
   Note:
   The drivers for most of the database connection types are automatically included and configured in Zend Studio. However, some driver types require you to manually add and configure the driver archive files.
   The following driver types come bundled and preconfigured in Zend Studio:
   - DB2
   - SB2 i-series
   - Derby
   - MySQL
   - Oracle

- Postre SQL
- SQL Server

5. Click Next.

   Select a driver from the drop-down list.

   If the required database driver has not been configured, click the New Driver Definition button to the right of the drop-down list and perform the following steps:

   i. In the Name/Type tab, select the required driver type and version.
   ii. Select the Jar List tab.
   iii. Click Add JAR/Zip.
   iv. Browse to the JAR/Zip file containing the required driver files and click Open.
   v. Click OK.

   The new driver is added to the driver definition list.

   vi. The properties for the selected driver are displayed.
   These will vary depending on the database type.
   vii. Click the Test Connection button to ensure all the details have been entered correctly.
   viii. Click Next to see a summary of your Connection Profile's details.
   ix. Click Finish.

Your new connection profile will be added to your databases list in the Data Source Explorer view.

You can now use this Connection Profile to connect to your database.

See the Connecting to a Database topic for more on how to connect to your database.

Note:

To change the properties of your connection profile, right-click it in the Data Source Explorer view and select 'Properties'.

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Connecting to a Database

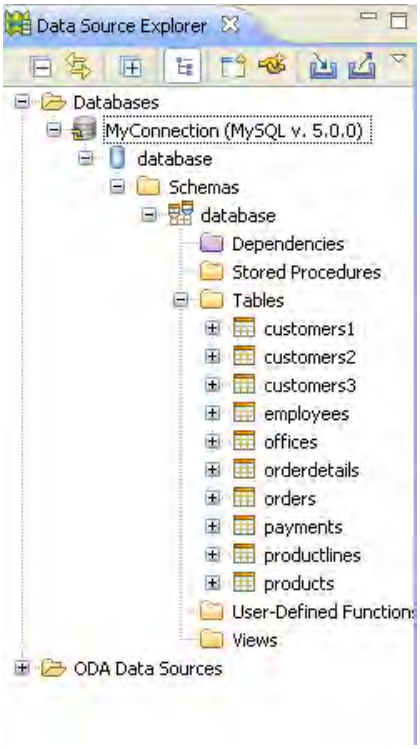
Once you have established your connection profile, you can connect to your database from the Data Source Explorer view.

This procedure describes how to connect to your database.

To connect to your database:

1. Open the Database Development perspective by going to Window | Open Perspective | Other | Database Development.
2. In the Data Source Explorer view, expand the SQL Databases node and right-click your connection profile.

   To edit the properties of the Connection Profile, right-click it and select Properties.
3. Click Connect.
4. Once the connection has been established, you can expand the tree underneath your Connection Profile to view the contents of the database.



To see a sample of the data in the tables, right-click one and select Data | Sample Contents. The SQL Results view will open with a sample list of the data from your table.

For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Note:

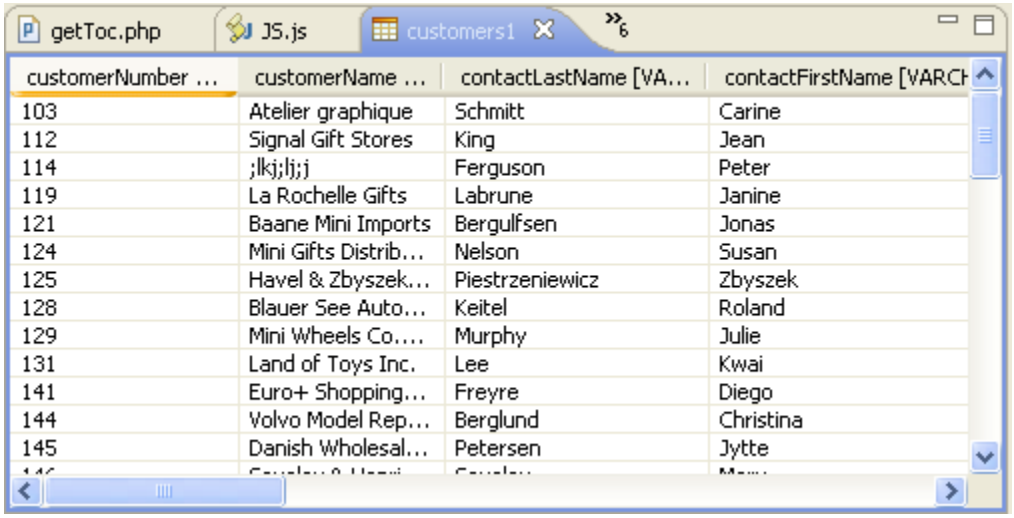
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Viewing and Editing Table Content

This procedure describes how to view data from a table located in your database and edit its contents.

To view and edit table content:

1. Open the Database Development perspective by going to Window | Open Perspective | Other | Database Development.
2. Connect to your Database by following the steps in the "Connecting to a Database" topic.
3. In Data Source Explorer view, double-click the required table or right-click it and select Data | Edit.
   The table will open in a database editor displaying all the data within the table.


getToc.php
JS.js
customers1
customerNumber ...
customerName ...
contactLastName [VA...
contactFirstName [VARCH
103 Atelier graphique Schmitt Carine
112 Signal Gift Stores King Jean
114 ;lkj;lj;j Ferguson Peter
119 La Rochelle Gifts Labrune Janine
121 Baane Mini Imports Bergulfsen Jonas
124 Mini Gifts Distrib... Nelson Susan
125 Havel & Zbyszek... Piestrzeniewicz Zbyszek
128 Blauer See Auto... Keitel Roland
129 Mini Wheels Co.... Murphy Julie
131 Land of Toys Inc. Lee Kwai
141 Euro+ Shopping... Freyre Diego
144 Volvo Model Rep... Berglund Christina
145 Danish Wholesal... Petersen Jytte

4. Select a cell to edit its contents.
5. Click Save .
   The changes made will be automatically applied to the database.

For more information on the Data Tools Platform, see the Data Tools Platform User Documentation.

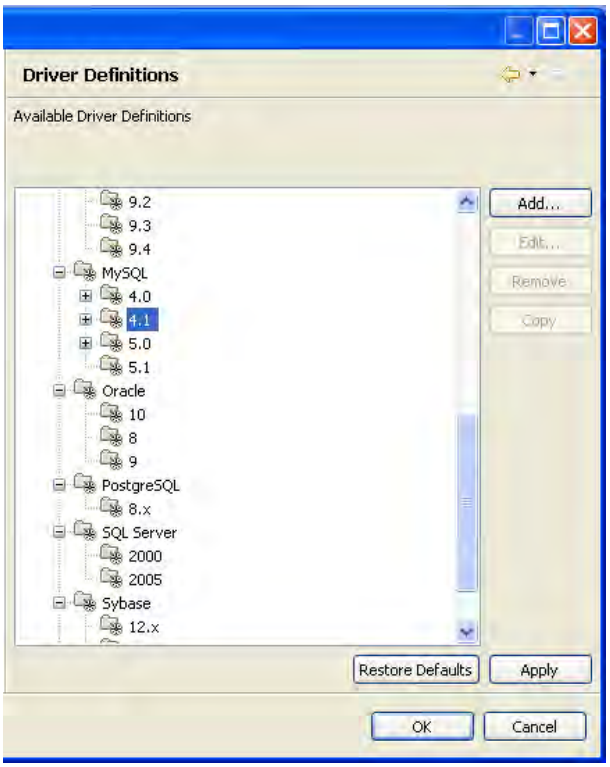
Creating a MYSQL Driver Definition

This procedure describes how to create a driver definition for your database. A driver definition maps to the location of your driver files.

You must have access to MySQL driver files before attempting to create a driver definition.

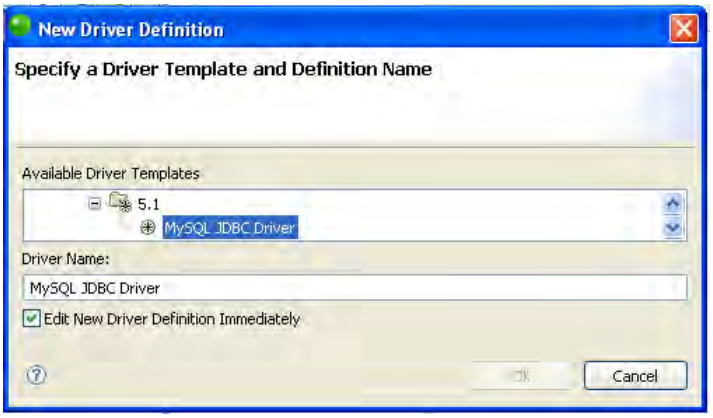
To create a driver definition for MySQL:

1. Open the Driver Definitions Preferences page by going to Window | Preferences | Connectivity | Driver Definitions.
2. A list of driver definitions will appear in the preferences page.
3. Scroll down to the MySQL category and select the required MySQL driver.



4. Click Add.

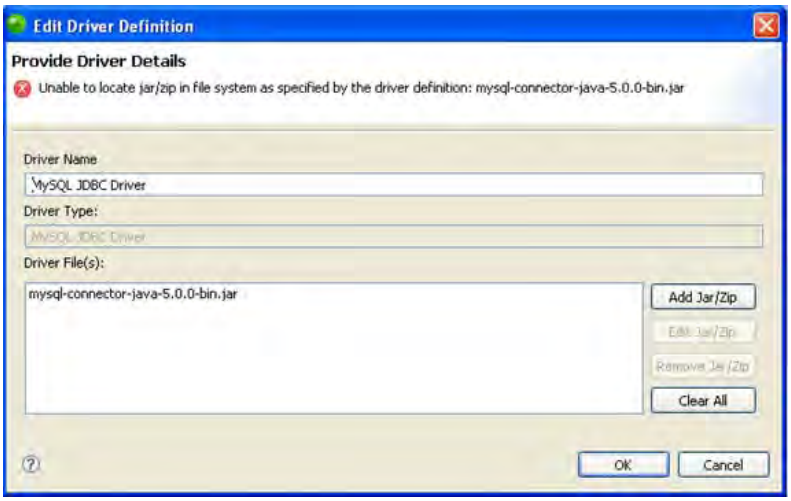
   The New Driver Definition wizard opens.



5. Expand the nodes until you see the MySQL JDBC Driver.
6. Select the driver.

   The default driver name will be entered in the Driver Name field. If required, delete this name and enter a new name.
7. Ensure that the "Edit New Driver Definition Immediately" checkbox is marked.
8. Click OK.

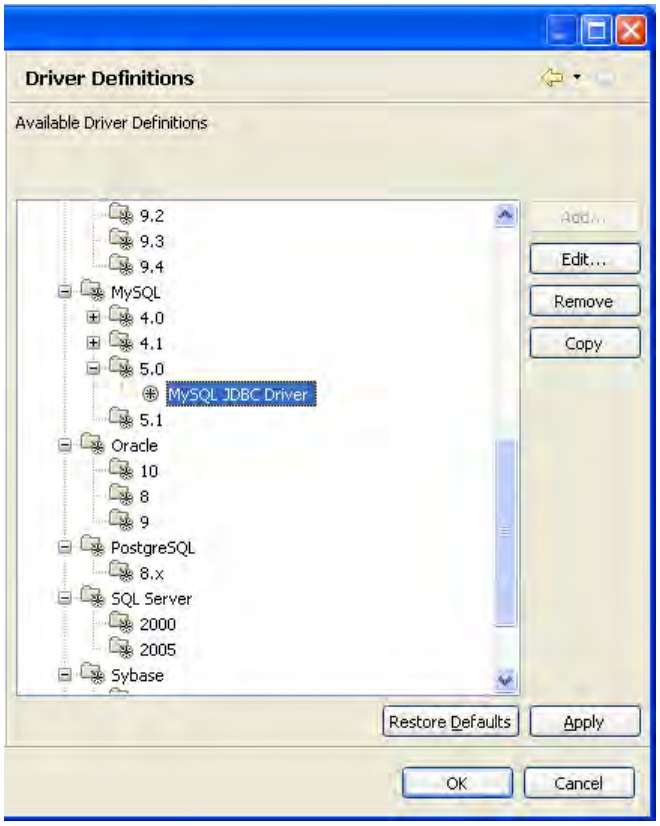
   The Driver Details wizard opens.



9. Delete the sample .jar driver file by selecting it and clicking Remove Jar/Zip.
10. Click Add Jar/Zip.
11. Browse your file system to find your MySQL driver and press OK.

    Your driver will be added to the list.
12. Click OK.

Your new driver definition will be added to your Driver Definitions list. This will allow you to access the driver when connecting to a MySQL database.


Driver Definitions
Available Driver Definitions
9.2
9.3
9.4
MySQL
4.0
4.1
5.0
MySQL JDBC Driver
5.1
Oracle
10
8
9
PostgreSQL
8.x
SQL Server
2000
2005
Sybase
Add...
Edit...
Remove
Copy
Restore Defaults
Apply
OK
Cancel

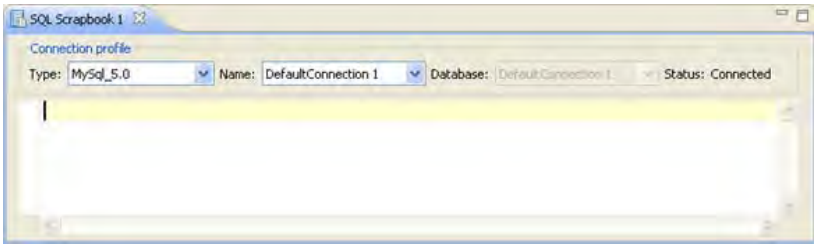
Creating and Executing an SQL Query

This procedure describes how you can run an SQL query on your database once you have created it.

You must have created a Connection Profile and connected to your database before using this functionality.

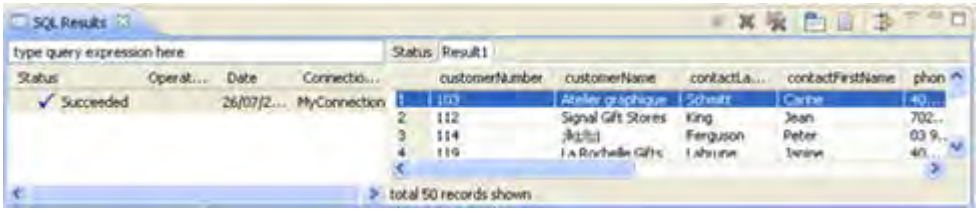
To run an SQL Query on your database:

1. Open the Database Development perspective by going to Window | Open Perspective | Other | Database Development.
2. Connect to your Database by following the steps in the "Connecting to a Database" topic.
3. Click the Open Scrapbook icon on the toolbar.
   A new SQL scrapbook will open.



4. Write your query in the scrapbook (e.g. select * from mytablename;)
5. To execute your query, right-click anywhere in the editor and select Execute All -or- press Ctrl+Alt+X.
   To execute only specified queries, highlight the relevant lines, right-click and select Execute Selected Text.

The query will be run and the results will be displayed in the Result1 tab in the SQL Results view. The left pane displays the execution history. For each statement that you execute, including stored procedures, an execution history entry is added to this pane. This allows you to quickly retest the execution using slightly different values and settings. You can rename or delete the launch configurations as needed.



For more information on the Data Tools Platform, please see the Data Tools Platform User Documentation.

Running Files and Applications

The following options are available when running your files and applications:

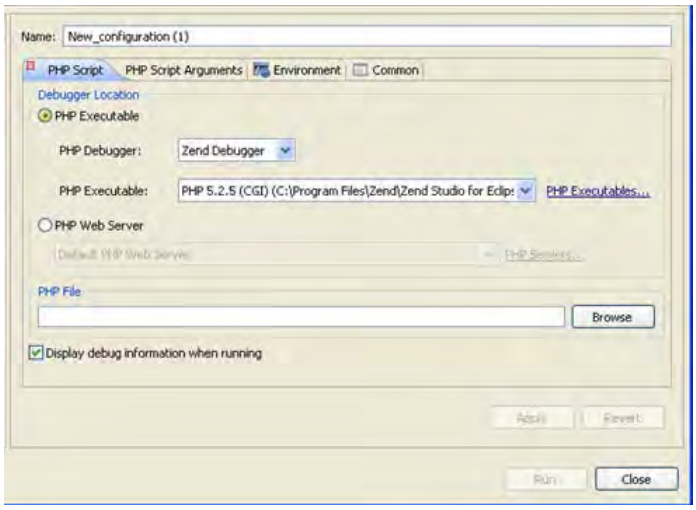
- Running PHP Scripts Locally - Run files on your workspace using Zend Studio's internal debugger.
- Running PHP Scripts Remotely - Run files on your workspace using your server's Zend Debugger.
- Running PHP Web Pages - Run applications situated on a server.

Running PHP Scripts Locally

This procedure describes how to run a PHP Script from your workspace using Zend Studio's internal debugger.

To locally run a PHP Script:

1. Click the arrow next to the Run button on the toolbar and select Run Configurations -or- go to Run | Run Configurations.
   A Run dialog will open.
2. Double-click the PHP Script option to create a new run configuration.



3. Enter a name for the new configuration.
4. Ensure that the PHP Executable option is selected under the Debugger Location category and select the required PHP executable.
5. Under PHP File, click Browse and select your file
6. Marking the 'Display debug information when running' checkbox will cause debug views to be displayed.
7. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
8. Click Apply and then Run.

Your script will be run and displayed in a browser.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Running PHP Scripts Remotely

This procedure describes how to run files in your workspace using your server's Zend debugger. Use this function if you want to test the execution of the file in 'real time' on the production server. This is especially relevant if your server has loaded extensions.

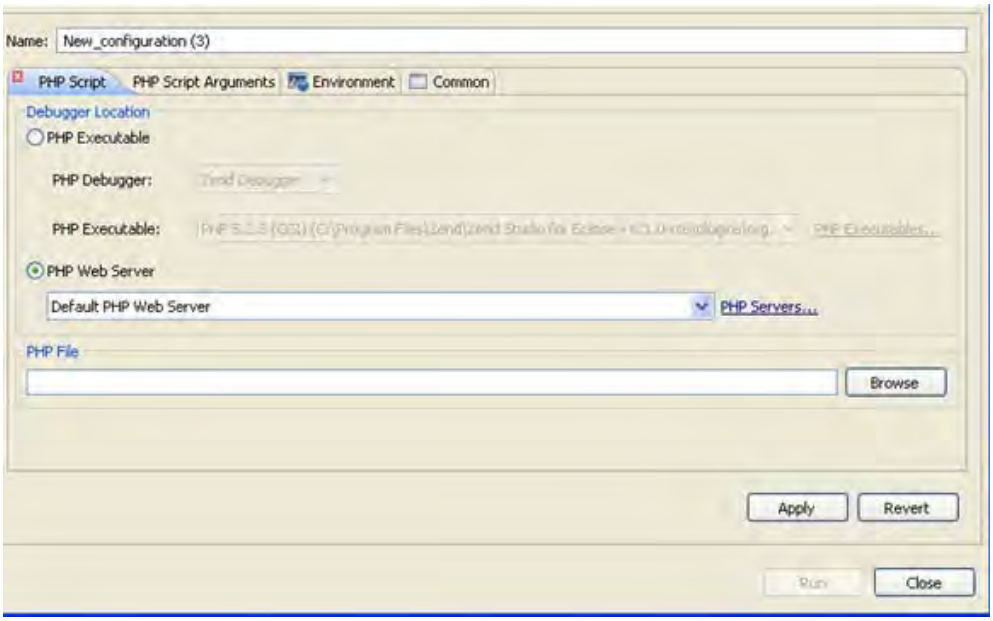
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To remotely run a PHP Script:

1. Click the arrow next to the Run button on the toolbar and select Run Configurations -or- go to **Run | Run Configurations**.
   A Run dialog will open.
2. Double-click the PHP Script option to create a new run configuration.



5. Enter a name for the new configuration.
6. Select the PHP Web Server option under the Debugger Location category.
7. Select your server from the list.

If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The Servers preferences page will open. Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page. For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.

8. Under the PHP File category, click Browse and select your file.
9. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
10. Click Apply and then Run.

Your script will be run and displayed in a browser.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.
In addition, if a file defined with an absolute path to a server location (See 'Include Paths' for more on absolute file locations) is called, a Path Mapping dialog will appear. See Path Mapping for more information.

Running PHP Web Pages

This procedure describes how to run whole applications, projects, files or collections of files that are on the server. Using this process, you can run either copies of the files which are located on the server or the files located locally on your workspace (if available).

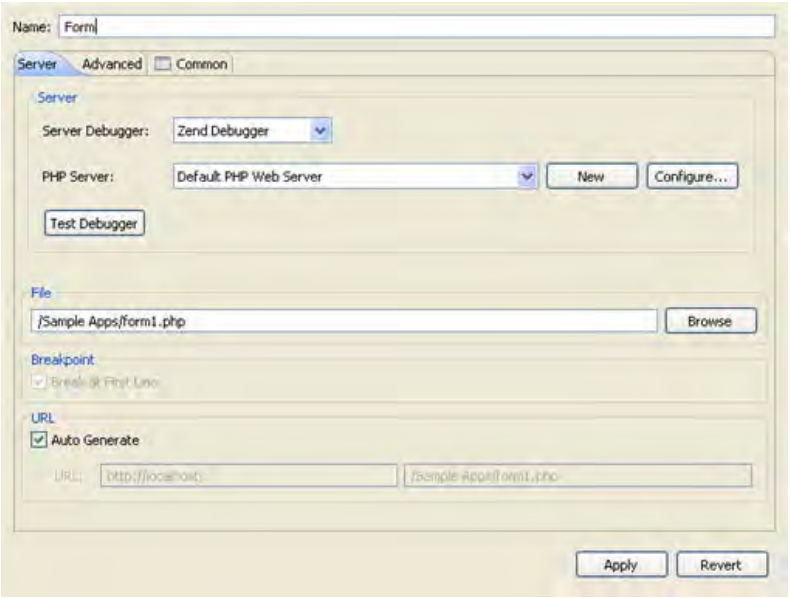
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To run a PHP Web Page:

1. Click the arrow next to the Run button on the toolbar and select Run Configurations -or- go to Run | Run Configurations.
   A Run dialog will open.
2. Double-click the PHP Web Page option to create a new run configuration.

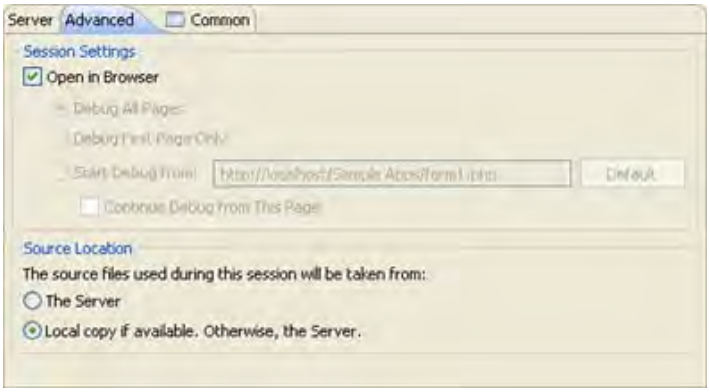


3. Enter a name for the new configuration.
4. Select the Server Debugger to be used (by default this will be the Zend Debugger). supports both the Zend Debugger and XDebug.

5. Select your server from the PHP Server list.
   If you have not yet configured your server, click New.
   The PHP Server Creation Wizard will open.
   Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page.
   For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.
6. To check whether your server connection is correctly configured and that Zend Studio can communicate with your server debugger, click the 'Test Debugger' button.
7. Under PHP File, click Browse and select the file which you would like to run.
8. The URL to be run will have been automatically created based on the file name and your server address. If the URL does not point to the file's location, unmark the Auto Generate checkbox and modify the URL.

Note:
The file to be run needs to exist on the server even if you are going to be selecting to run the local copy of your files.

9. For further options, select the Advanced tab.



9. Under the Source Location category you can choose whether the content of the files to be run will be taken from the server or from your workspace.
   If a local copy is not available, files will be taken from the server. Selecting the 'Local Copy' option will result in the Path Mapping mechanism being applied when files are called. See the Path Mapping topic for more details.

Note:
The file to be run needs to exist on the server even if you are going to be selecting to run the local copy of your files.

10. Click Apply and Run.

Your application will be run and displayed in a browser.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment. In addition, if a file defined with an absolute path to a server location (See 'Include Paths' for more on absolute file locations) is called, a Path Mapping dialog will appear. See Path Mapping for more information.

Debugging Files and Applications

The following Debug functionality is available in Zend Studio:

- Local PHP Script Debugging - Debug files on your workspace using Zend Studio's internal debugger.
- Remote PHP Script Debugging - Debug files on your workspace using your server's Zend Debugger.
- PHP Web Page Debugging - Debug applications situated on a server.
- URL Debugging - Enter a URL to debug an application on a server.
- Debugging Using the Zend Debugger Toolbar - Debug files and applications directly from your browser.

Once a debug session has been launched, the PHP Debug perspective is used to control the debugging process and to view and analyze the results.

See the "Running and Analyzing Debugger results" topic for more information on controlling and monitoring the debugging process.

Setting Breakpoints

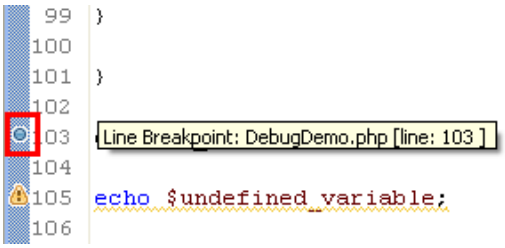
Before you debug your scripts, you can set breakpoints in them to specify places in your code where the debugging process will pause.

These procedures demonstrate how to set line and conditional PHP breakpoints.

To set a breakpoint in your script:

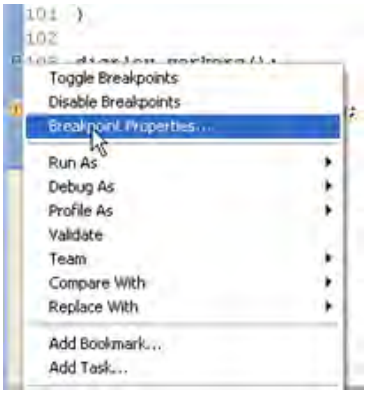
Double-click the vertical ruler to the left of the line where you want to set the breakpoint -or- select the line and go to Run | Toggle Breakpoint or press Ctrl+Shift+B.

A blue ball will appear, indicating that a breakpoint has been set.



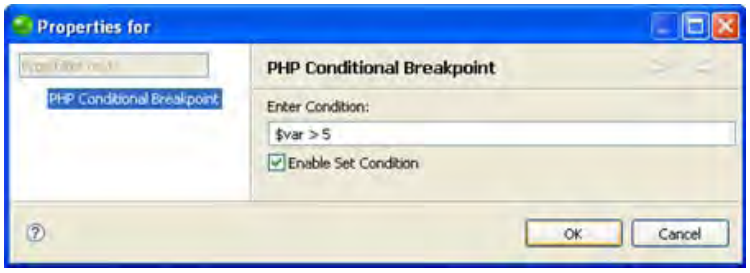
To add a condition to a breakpoint:

1. Right-click the breakpoint in the vertical marker bar and select Breakpoint Properties...



The PHP Conditional Breakpoint dialog opens.

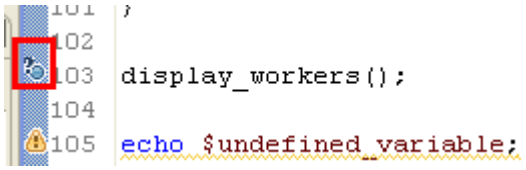
2. Enter the required condition and ensure the Enable Set Condition checkbox is marked.



3. Click OK.

The condition will be set for the breakpoint and the debugging process will pause only if the condition is met.

The breakpoint will be represented by a blue ball with a question mark.



Debugging a PHP Script

Files located in your workspace can be debugged in two ways:

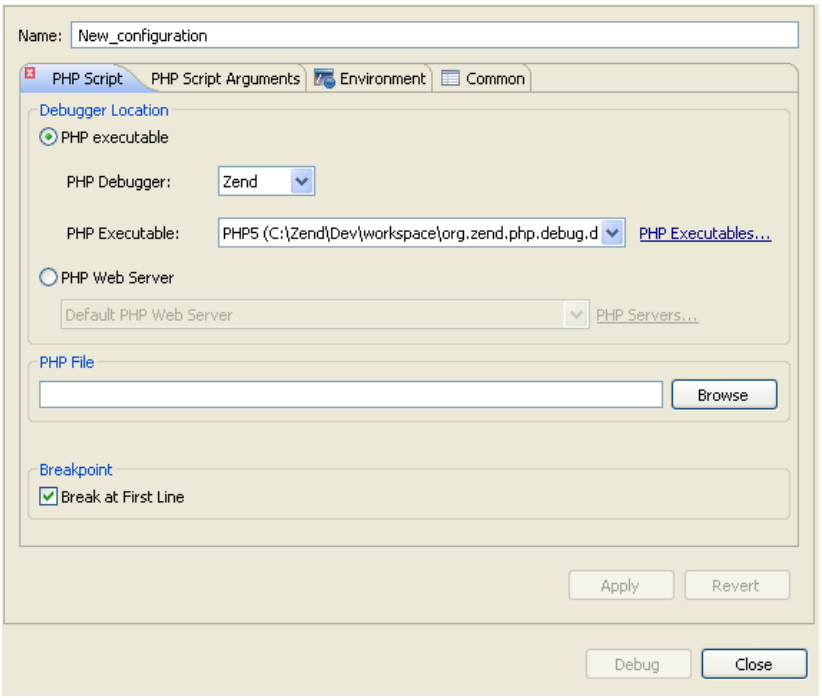
- Local Debugging - Using Zend Studio's internal debugger
- Remote Debugging - Using your remote server's debugger

Locally Debugging a PHP Script

This procedure describes how to debug a PHP Script from your workspace using an internal PHP Executable.

To locally debug a PHP Script:

1. Set breakpoints at the relevant places in the file that you would like to debug by double-clicking the vertical marker bar to the left of the editor.
2. Save the file.
3. Click the arrow next to the debug button on the toolbar and select Debug Configurations... -or- select **Run | Debug Configurations....**
   A Debug dialog will open.
4. Double-click the PHP Script option to create a new debug configuration.



5. Enter a name for the new configuration.
6. Ensure that the "PHP Executable" option is selected under the Debugger Location category and select the required PHP executable.
7. Select the Zend Debugger from the PHP Debbuger list.
8. Enter your PHP file in the "PHP File" text field, or click **Browse** and select your file
9. Marking the "Breakpoint" checkbox will result in the debugging process pausing at the first line of code.

10. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
11. Click **Apply** and then **Debug**.
12. Click **Yes** if asked whether to open the PHP Debug Perspective.

A number of views will open with relevant debug information.
See the Running and Analyzing Debugger results topic for more information on the outcome of a debugging process.

Note:
If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Remotely Debugging a PHP Script

This procedure describes how to debug files in your workspace remotely using your server's Zend Debugger. Use this function if you want to test the execution of the file in 'real time' on the production server. This is especially relevant if your server has loaded extensions.

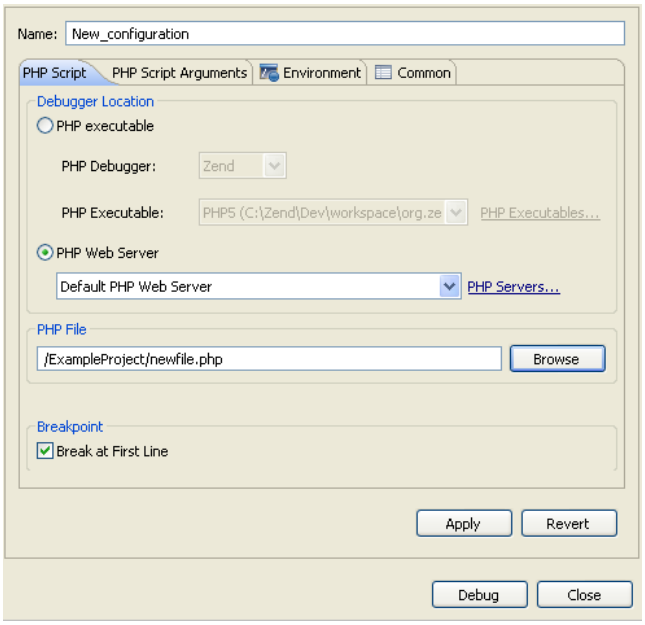
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To remotely debug a PHP Script:

1. Set breakpoints at the relevant places in the file that you would like to debug.
2. Save the file.
3. Click the arrow next to the debug button on the toolbar and select Debug Configurations... -or- select Run | Debug Configurations....
   A Debug dialog will open.
4. Double-click the PHP Script option to create a new debug configuration.



5. Enter a name for the new configuration.

6. Select the PHP Web Server option under the Debugger Location category.
7. Select your server from the list.
   If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The Servers preferences page will open. Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page. For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.
8. Under the PHP File category, click Browse and select your file.
9. Marking the Breakpoint checkbox will result in the debugging process pausing at the first line of code.
10. If necessary, you can add arguments in the PHP Script Arguments tab to simulate command line inputs.
11. Click Apply and then Debug.
12. Click Yes if asked whether to open the PHP Debug Perspective.

A number of views will open with relevant debug information.
See the "Running and Analyzing Debugger results" topic for more information on the outcome of a debugging process.

Note:
If the remote debugging session is unsuccessful, check that your server's root directory contains a Dummy File. This file should match the name of the Dummy File as defined in the Advanced Options section of the PHP Debug preferences page (accessible from Window | Preferences | PHP | Debug). By default, the file will be called "dummy.php".

Note:
If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.
In addition, if a file defined with an absolute path to a server location (See 'Include Paths' for more on absolute file locations) is called, a Path Mapping dialog will appear. See Path Mapping for more information.

Debugging a PHP Web Page

This procedure describes how to debug whole applications, projects, files or collections of files that are already on the server.

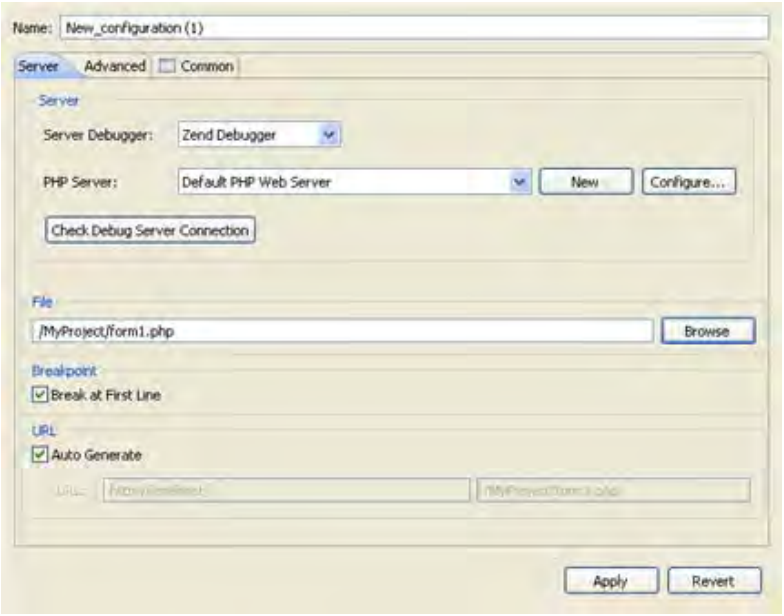
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To debug a PHP Web Page:

1. Click the arrow next to the debug button on the toolbar and select Open Debug Dialog -or- select Run | Open Debug Dialog.
   A Debug dialog will open.
2. Double-click the PHP Web Page option to create a new debug configuration.



3. Enter a name for the new configuration.
4. Select the Server Debugger to be used (by default this will be the Zend Debugger).
5. Select your server from the PHP Server list.
   If you have not yet configured your server, click New.
   The PHP Server Creation Wizard will open.
   Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page.
   For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.
6. To check whether your server connection is correctly configured, and that Zend Studio can communicate with your server debugger, click the 'Check Debug Server Connection' button.
7. Under PHP File, click Browse and select your 'debug target' file (the file from which the debugging process will start.)
8. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' checkbox.
9. The URL to be debugged will have been automatically created based on the file

name and your server address. If the URL does not point to your debug target's location, unmark the Auto Generate checkbox and modify the URL.

10. For further Debug options, select the Advanced tab, which has the following options:

New Debug Configuration - Advanced

- Open in Browser - Mark if you would like the application to be displayed in Zend Studio's internal browser while it is debugged.
- Choose whether to:
    - Debug All Pages - The specified page and all the pages linked to it are debugged. The browser waits for the debug of each page before displaying it.
    - Debug First Page Only - Only the first page is debugged.
    - Start Debug from - Select the URL from which you would like the Debugging process to start.
    - Continue Debug from this Page - Selecting this option will result in all the pages linked to the URL being debugged.
- Source Location - Choose whether the source files used during this session will be taken from the server or from a local copy.
  If a local copy is not available, files will be taken from the server. Selecting the 'Local Copy' option will result in the Path Mapping mechanism being applied when files are called. See the Path Mapping topic for more details.

10. Click Apply and then Debug.
11. Click Yes if asked whether to open the PHP Debug Perspective.

See the "Running and Analyzing Debugger results" topic for more information on the outcome of a debugging process.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.
In addition, if a file defined with an absolute path to a server location (See 'Include Paths' for more on absolute file locations) is called, a Path Mapping dialog will appear. See Path Mapping for more information.

Debugging a URL

This procedure describes how to debug a URL on a server to which you have access.

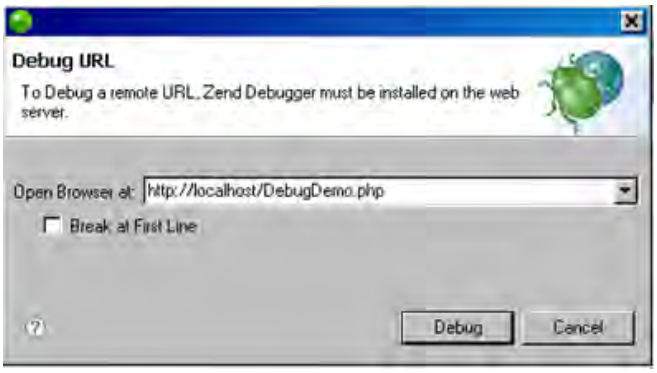
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To debug a URL:

1. Click the Debug URL button on the main toolbar -or- go to Run | Debug URL.
2. The Debug URL dialog will appear.



3. In the 'Open Browser at' field, enter the URL of the first page that should be debugged.
4. Select whether the Debugger should stop at the first line of code by marking/unmarking the 'Break at First Line' checkbox.
5. Click Debug.

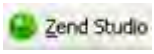
The Debug Perspective will open with a number of views detailing information about the debugging process.

See the "Running and Analyzing Debugger results" topic for more information on the outcome of a debugging process.

This procedure describes how to debug using the Zend Browser Toolbar.

Note:
In order to enable Zend Studio to debug files located on the server, remote debugging needs to be configured. See Setting Up Remote Debugging for more details.

To debug using the Zend Browser Toolbar:

1. Ensure the Zend Browser Toolbar is installed on your browser.
   If you have not yet installed the Zend Browser Toolbar, see Installing and Configuring the Zend Browser Toolbar for more information.
2. Open your browser and browse to the page from which you would like to start debugging.
3. Ensure Zend Studio is open. You can open it by clicking the Zend Studio button 

in the toolbar.
4. Click the

button on the Zend Browser Toolbar to debug the page currently displayed in the browser.
   To do this, the page is reloaded, and instructions are automatically transmitted to the server for the execution to be done in debug mode. This means that if the POST data has been transmitted to the current page, the browser will ask if the user wishes to post them again before executing the page in debug mode. In the same way, if the page uses frames (including hidden ones), the toolbar will request which frame the user wants to debug.
   -Or- select one of the following debug options by clicking the arrow to the right of the Debug button :
   - Next page on site - The debugging session will be launched when the next link is clicked, a form is posted, or an AJAX request is executed.
   - All forms (POST) on this site - The debugging session will be launched every time a link is clicked, a form is posted or an AJAX request is executed using the POST method. The script that will be debugged will be the script designated as the action of the form or link.
   - All pages on this site - Debugs all pages from the current page.

Note:

If the files you would like to debug exist in your workspace, you can choose to debug the workspace copy of your files by going to Extra Stuff | Settings on the Toolbar and selecting the 'Debug Local Copy' option.

In Zend Studio for Eclipse, if path mapping has not yet been configured, a path mapping dialog will be displayed once the debugging session is launched to determine which workspace files will be debugged. See Path Mapping for more details.

The relevant debug session will be launched in Zend Studio.

Installing and Configuring the Zend Browser Toolbar

Installing the Zend Browser Toolbar

The Zend Browser Toolbar can be installed during the installation of Zend Studio or can be downloaded and installed separately.

To install the Zend Browser Toolbar during Zend Studio installation:

In the 'Choose Install Set' dialog of the Zend Studio installation, select the Zend Firefox Toolbar and/or the Zend Internet Explorer Toolbar options:

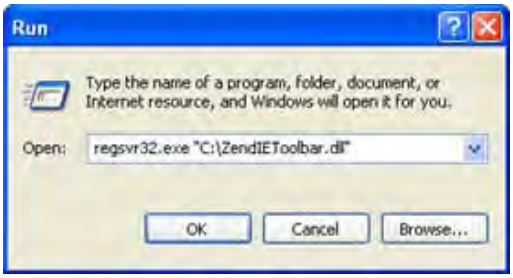
Choose Install Set
Install Set
Full
Zend Studio
Zend Firefox Toolbar
Zend Internet Explorer Toolbar
Description
Zend Studio is a powerful integrated development environment for developing and maintaining PHP applications. Zend Studio combines a powerful editor, code inspectors and analyzers, a debugger and profiler, and robust code completion engine for all PHP constructs as well as HTML.
Configuration
InstallAnywhere by Macrovision
Cancel
Previous
Next

To manually install the Zend Internet Explorer Toolbar:

1. Go to the Zend Studio downloads site (http://www.zend.com/en/products/studio/downloads) and download the Studio Browser Toolbar for Internet Explorer on Windows.
2. Save the file to your file system.
3. Run the following command (by going to Start | Run in the Windows Start Menu):

```
regsvr32.exe "<ZendIEToolbar.dll_Location>\ZendIEToolbar.dll"
```

Replace <ZendIEToolbar.dll_Location> with the path to the directory in which you saved the downloaded file.



4. A dialog will be displayed confirming the successful registration of the .dll.
5. Restart Internet Explorer.
6. If the Toolbar is not automatically displayed, select the Zend Studio toolbar from the Internet Explorer Toolbars list.

Note:
In Internet Explorer 7, the Toolbar list is available from Tools | Toolbars.

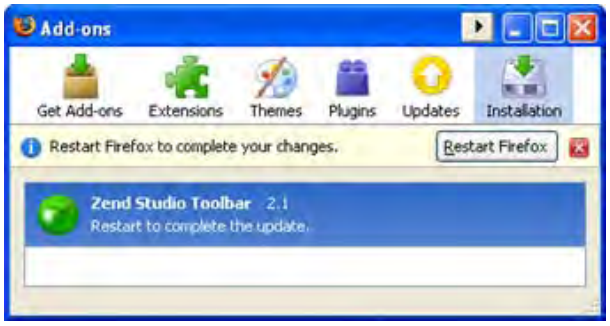
You should now configure the Zend Studio communication settings for the Zend Browser Toolbar in order to be able to debug/profile.

To manually install the Zend Firefox Toolbar:

1. Go to the Zend Studio downloads site (http://www.zend.com/en/products/studio/downloads) and download the cross-platform Firefox Browser Toolbar.
2. Open the downloaded .xpi file by going to FIle | Open File from the Firefox menu bar and browsing to the downloaded file.
3. A prompt will be displayed asking whether you trust the add-on.

Software Installation
Only install add-ons from authors whom you trust.
Malicious software can damage your computer or violate your privacy.
You have asked to install the following item:
ZendFirefoxToolbar-2.1.xpi (Author not verified)
Cancel

4. Click Install Now.
   The add-on will be displayed in the Firefox Add-ons list.



5. Click the Restart Firefox button.
   Firefox will be restarted and the toolbar will be loaded.
6. If the Toolbar is not automatically displayed, select the Zend Studio toolbar from Firefox's Toolbars list.

Note:
In Firefox 3, the Toolbar list is available from View | Toolbars.

You should now configure the Zend Studio communication settings for the Zend Browser Toolbar in order to be able to debug/profile.

Configuring the Zend Browser Toolbar

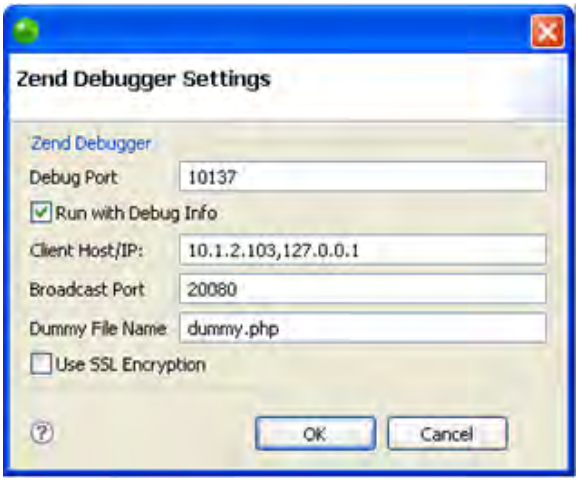
In order to be able to debug files and applications through the Zend Browser Toolbar, you must configure the toolbar to communicate with Zend Studio.
Toolbar debugging is a form of remote debugging and so remote debugging to the server on which your files are located must also be configured in Zend Studio and the Zend Debugger. See Setting Up Remote Debugging for more details.

This procedure describes how to configure the Zend Browser Toolbar to be able to communicate with Zend Studio.

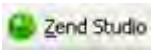
To configure connectivity with Zend Studio:

1. From the Toolbar, go to Extra Stuff | Settings.
   The Zend Toolbar Settings dialog will be displayed.

2. In the Zend Studio Settings category, configure the Zend Studio Debug Port and Zend Studio IP Address. These settings should match the settings defined for the Zend Debugger in the Installed Debuggers Preferences page.



To automatically synchronize your Zend Browser Toolbar settings with your Zend Studio settings:

1. Ensure Zend Studio is open.
   If your Zend Studio executable is configured in the Zend Toolbar Settings dialog, you can launch Zend Studio by clicking the Zend Studio button 

in the toolbar.
2. In the Zend Studio Settings category, select 'Auto Detect Settings'.
3. Enter the Zend Studio Broadcasting Port. This must match the Broadcasting Port configured in your Zend Studio debug preferences.
   The default port is 20080.
4. Click Test.
   The Debug Port and Zend Studio IP settings defined in Zend Studio will be automatically updated in your Zend Browser Toolbar and a message will appear confirming that the Auto Detect test was completed successfully.

To manually enter your settings:

1. Select 'Manual Settings'.
2. Configure the following settings:

- Zend Studio Debug Port - The debug port to be used for the debug session. This should match the Debug Port setting configured in your Zend Studio debug preferences.
- Zend Studio IP Address - Enter the Client Host/IP address of the machine on which your Zend Studio is installed. This should be the same machine from

which you are launching your debug session.
This should match the Client Host/IP setting configured in your Zend Studio debug preferences.

3. Click OK to save your settings.

Additional Configuration Options

Additional configuration options can be accessed through the Zend Studio Toolbar Settings dialog (Extra Stuff | Settings) as follows:

- **General:**
  - Enable Search Toolbar - Displays the PHP information search box in the Zend Browser Toolbar.
- **Debug Session Settings:**
  - Detect Zend Debugger (only available in the Zend Internet Explorer Toolbar) - disables the Debug and Profile buttons when a Zend Debugger is not detected on the server.
  - Debug Local Copy - Choose whether the source files used during the debug session will be taken from the server or from a local workspace copy of the files in Zend Studio (if available). If a local copy is not available, files will be taken from the server.

Note:
Enabling this option will result in the Path Mapping mechanism being applied when files are called. See Path Mapping for more details.

  - Break on the first line - this will cause the debugger to pause at the first line of PHP code during debugging.
- **Zend Studio Settings:**
  - Zend Studio Executable - Browse to the location of your Zend Studio executable file. This will enable you to launch Zend Studio by clicking the Zend Studio button

in the toolbar.

Running and Analyzing Debugger Results

Once you have launched one of the debug sessions (PHP Script, PHP Web Page or URL), you can control and monitor the debugging process using the views displayed in the debugging process.

Controlling the debugging process:

The debug process can be controlled using the various buttons in the Debug view.

1. The Debug process will automatically stop at each breakpoint.
2. The various views will display information about the debugging process up to that point only.
3. You can use the various buttons in the debug view to decide how to continue with the debugging process:

- Click the Resume button to continue the debugging process until the next breakpoint, or until the end of the debugging process.
- Click the Terminate button to stop the debugging process.
- Click the Step Over button to step over the next method call (without entering it) at the currently executing line of code. The method will still be executed.
- Click the Step Return button to return from a method which has been stepped into. The remainder of the code that was skipped by returning is still executed.
- Click the Step Into button to step into the next method call at the currently executing line of code.
- Click the Use Step Filters button to change whether step filters should be used in the current Debug View.
- Once the debugging process has terminated, you can click the Remove Terminated Launches button to remove any terminated debug sessions from the list.

During the debugging process (for PHP Script, PHP Web Page or URL), various views will provide the following information:

- Debug View - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out off) certain functions.
- Variables - Will display the various variables in your script.
- Breakpoints - Will display the breakpoints you have entered
- Parameter Stack - Will display the parameters through which functions are reached.

- Editor Window - Will display the code at the relevant sections, according to which line is selected in the Debug View window.
- Debug Output - Will show the textual output of the script. This will be updated as the debugging process continues.
- Browser Output - Will show the output of the script to a browser. This will be updated as the debugging process continues.
- Console View (External Link) - Displays any error and warning messages.
- Tasks View (External Link) - Displays tasks that were added to your script (if applicable).

Note:

See 'PHP Debug Perspective' for more on the views that will be displayed during Debugging.

Setting Up Remote Debugging

Before debugging on a server, whether using remote PHP Script or PHP Web Page debugging, certain settings need to be configured to ensure that Zend Studio can communicate with your server.

To set up communication between Zend Studio and the server on which you are debugging:

1. Ensure the Zend Debugger is installed on your server.
   The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.
2. Ensure the machine on which your Zend Studio is installed is an allowed host for your debugger.
   The method for configuring this setting will depend on whether you have Zend Server, Zend Platform, Zend Core, or the standalone Zend Debugger installed on your system.
   See Setting your Zend Studio to be an Allowed Host for more information.
3. In Zend Studio, configure your server according to the instructions under Adding Servers in the PHP Servers Preferences.
4. Ensure the correct settings are configured in your Debug Preferences and Installed Debuggers Preferences pages.
5. Ensure you have a dummy.php file in your remote server's document root.
6. If your server is situated behind a firewall or other security device, see Setting Up Tunneling for information on how to enable a communication tunnel.
   If you don't know whether your server is situated behind a firewall, contact your System Administrator.

Setting your Environment to be an Allowed Host

This procedure describes how to ensure the machine on which your Zend Studio is installed will be an Allowed Host for initiating your Debug and Tunneling session on the remote server.

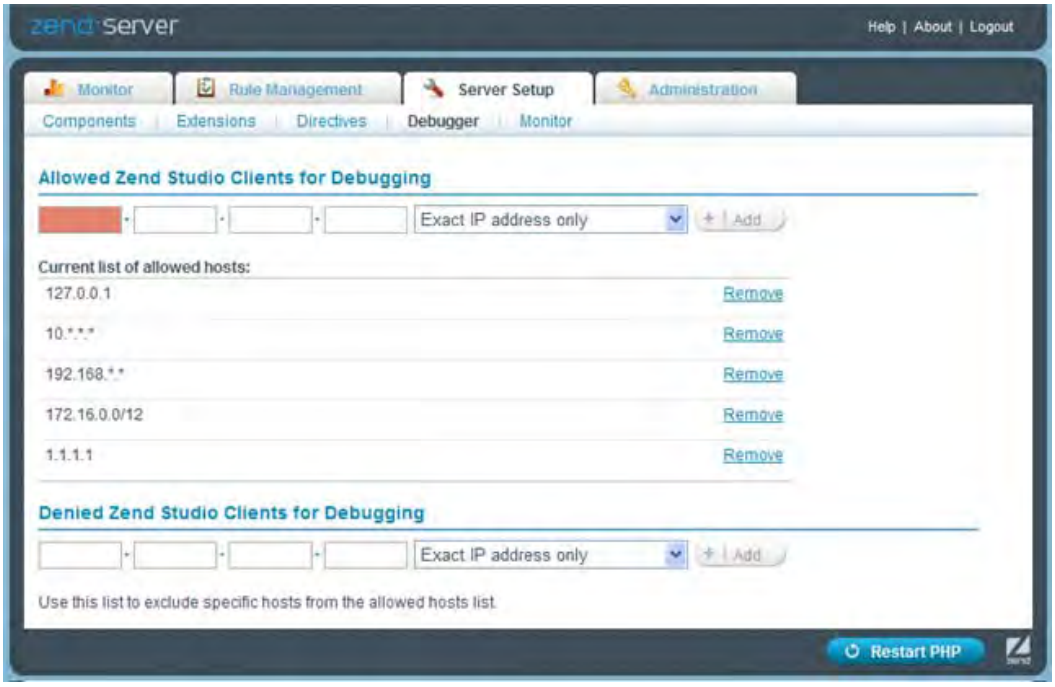
The steps you need to follow will depend on whether you have Zend Server or the standalone Zend Debugger installed on your server:

If Zend Server is installed:

1. Open your Zend Server GUI.

Note:
This can be done from within Zend Studio by selecting the server on which you have configured your Zend Server integration from the drop-down list next to the Zend Server icon on the toolbar .

2. Go to the Server Setup | Debugger tab.



3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections. To add an address to the list:
   i. Under the Allowed Zend Studio Clients for Debugging category, enter the IP of the machine on which your Zend Studio is installed.

ii. Click Add.

The IP Address is added to the Allowed Hosts list.

iii. Ensure your Studio's IP address is not in the Denied Hosts list.

If it is, click Remove next to the required address to remove it from the list.

iv. Click the

button to apply your settings.

If only the standalone Zend Debugger is installed on your server:

To configure your debugger to allow your Zend Studio to debug:

1. Open your php.ini file.
2. Edit the zend_debugger.allow_hosts and zend_debugger.allow_tunnel (if necessary) parameters to include the IP address of the machine on which your Zend Studio is installed.
   e.g. zend_debugger.allow_hosts=127.0.0.1/32
   zend_debugger.allow_tunnel=127.0.0.1/32
3. Ensure the address is not in your zend_debugger.deny_hosts parameter list.
4. Set the Debug Server to expose itself to remote clients by setting the zend_debugger.expose_remotely parameter to Always.
   (e.g. zend_debugger.expose_remotely=always).
5. Save the file.
6. Restart your Web server for the settings to take effect.

Ensuring the Placement of dummy.php

In order for the remote server's debugger to communicate with Zend Studio, a file called dummy.php must be located in your server's document root.

With the default Zend Server installation, a dummy.php file will have been automatically placed in your server's document root folder.
If you installed the standalone Zend Debugger, you must copy the dummy.php file from the Zend Debugger archive to your server's document root.

Note:
If you have set up a virtual host and it's document root is not pointed at the remote server's default document root, you will have to copy the dummy.php to the virtual host's document root in order to be able to debug on that virtual host.

You must also ensure that the Dummy File name is set to 'dummy.php' in Zend Studio.

To check your dummy file configuration in Zend Studio:

1. Open the Installed Debugger Preferences page by going to Windows| Preferences | PHP | Debug | Installed Debuggers.
2. In the Installed Debuggers list, select the Zend Debugger and click Configure.
   The Zend Debugger Settings dialog will open.
3. In the Dummy File Name setting, ensure 'dummy.php' is entered.

   Note:
   If you changed the name of the dummy file on the server, you must change this entry accordingly.

4. Click OK.

Adding a Server Location Path Map

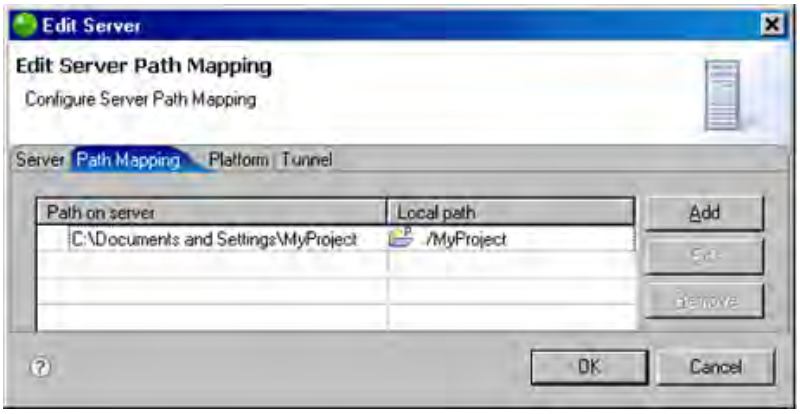
This procedure describes how to add a Path Map to a server so that files which are called from a certain location on the server will be searched for in a local location during remote PHP Script debbuging/profiling and PHP Web Page debugging/profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).
See Path Mapping for more details

To add a Path Map to a server:

1. Open the PHP Servers Preferences Page by going to Window | Preferences on the Menu Bar and selecting PHP | PHP Servers from the Preferences list.
2. Select the server on which you would like to create the Path Map and click Edit.
3. In the Edit Server dialog, select the Path Mapping tab.
4. Click Add.
5. An Add new Path Mapping dialog appears.
6. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.
7. Select either the 'Path in Workspace' or 'Path in File System' option and click Browse to specify the location.

8. Click OK.

Your Path Map will be added to your server list.



The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

Note:

Path Mapping can also be set automatically during Debugging / Profiling. See the Path Mapping topic for more details.

Setting Up Tunneling

To establish a tunneling connection for remote debugging and Zend Server integration, the following tasks need to be performed:

Note:

If you have followed the instructions under Setting Up Remote Debugging, you will have already performed some of these tasks.

1. Ensure you have a dummy.php file in your remote server's document root.
2. Ensure your Zend Studio is an allowed host for your server debugger. This can be done through Zend Server, Zend Core or Zend Platform (if installed), or through your php.ini file.
3. Configuring you Tunneling Connection in Zend Studio - From the PHP Servers Preferences page (Window | Preferences | PHP | PHP Servers).
4. Activate the tunnel - By selecting your Tunneling server from the list next to the Tunneling icon on the toolbar.

Configuring Zend Server to Auto Detect Studio Settings

This procedure describes how to configure Zend Server so that Zend Studio's settings are automatically detected during the Debugging / Profiling of Zend Server events.

To establish a communication tunnel between Zend Server and Zend Studio:

1. Open your Zend Server GUI.
2. Go to the Server Setup | Monitor tab.
3. In the Zend Server Settings section, configure the following:
   - Auto detect the Zend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to detect your Zend Studio Debug settings.
4. Click Save.

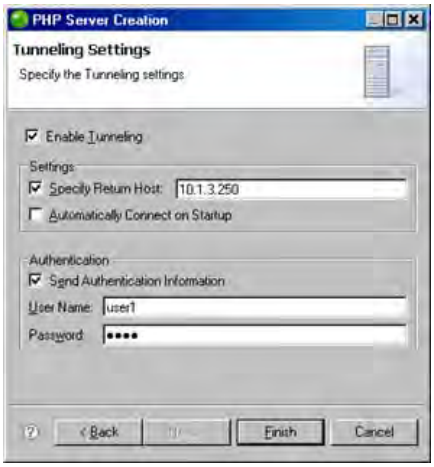
Setting Up a Tunneling Server

This procedure describes how to configure a server to allow Tunneling in Zend Studio.

Note:
You can configure several servers to allow tunneling.

To configure your server for Tunneling with Zend Studio:

1. Open the PHP Server Preferences page by going to Window | Preferences | PHP | PHP Servers from the Menu Bar.
2. Click New to define a New Server (or Edit if the server has already been defined).
3. Give the server a unique name and enter the URL of the server to which you would like to create a tunnel.
   Click Next to continue or go to the next Tab.
4. You can ignore the Path Mapping option. If necessary, you will be automatically prompted to define path mapping during debugging and profiling sessions. See "Path Mapping" for more details.
   Click Next to continue or go to the next Tab.
5. If you want to enable Zend Server integration mark the Enable Zend Server Integration checkbox.
   See Configuring Zend Server Settings in Zend Studio for more information.
6. Click Next to continue or go to the next Tab.



7. In the Tunneling Settings section, check the "Enable Tunneling" option.
8. In some system configurations, a dedicated tunneling server is used which allows debug connectivity between Zend Studio and the server which is being debugged. In

this case, unmark the 'Specify Return Host' checkbox and enter the IP address of the tunneling server to which the tunneling connection will be opened.

9. If your Web server requires HTTP authentication, enter your User Name and Password in the Authentication category. Zend Studio sends the authentication information in the header.

> **Note:**
> This assumes the user account is set up on the Web server

10. Click Finish or OK.

Your server is now configured to allow tunneling in Zend Studio.

The next step is to Activate your Tunneling Connection.

Activating Tunneling

Once you have configured all the necessary settings in Zend Studio and on your server, you can activate your Tunnel connection.

This procedure describes how to open a tunnel between Zend Studio and your remote server.

To activate Tunneling in Zend Studio:

1. Click the arrow next to the Tunneling icon on the toolbar and select the server which you configured for tunneling.

2. The tunneling icon will turn green to show that a tunneling server is connected:

You can now debug/profile on the selected remote server.

Note:

Several Tunneling sessions can be configured in Zend Studio. Therefore, If the debug session is not working, check to see that the Tunnel to the correct server is connected by clicking on the drop-down arrow next to the Tunneling Icon and verifying that the name of the connected server is correct.

Troubleshooting the Communication Tunnel

If Zend Studio is unable to connect to the target server, you will get an error message with the response from the server. The table below describes the most likely causes and a recommended action for successfully establishing a connection with the target server.

Possible Cause	Recommended Action
The server address you entered is incorrect	Enter the correct server information in the Tunneling Settings dialog.
HTTP authentication is required	Enter authentication information in the Tunneling Settings dialog box; then click the 'Send authentication information' checkbox.
The dummy file content or location on the server is incorrect	The dummy file on the server side was either changed or does not exist. You will need to ensure that the correct dummy file with the correct content is placed in the correct directory on the target server. The correct dummy file is created and located properly as part of the installation procedure. The problem here is post-installation.
You are not allowed to connect with the server via the communication tunnel	You must have tunneling permissions in the php.ini file. Make sure that the zend_debugger.allow_tunnel variable is properly configured. For any other cause, or additional information, use one of our support options.

Profiling Files and Applications

The following Profile functionality is available in Zend Studio:

- Profiling a PHP Script - Profile PHP Scripts on your Workspace
- Locally Profiling a PHP Script - Profile a PHP Script using Zend Studio's internal debugger
- Remotely Profiling a PHP Script - Profile a PHP Script using your remote server's debugger
- Profiling a PHP Web Page - Profile PHP files on a remote server
- Profiling a URL - Profile a URL
- Profiling Using the Zend Debugger Toolbar - Profile directly from your web browser.

Once a Profile session has been launched, the PHP Profile perspective is used to view and analyze the results of the profiling process.

See the PHP Profile Perspective topic for more on the information displayed once a Profile session has been run.

Profiling a PHP Script

Files located in your workspace can be profiled in two ways:

Locally Profiling a PHP Script - Using Zend Studio's internal debugger

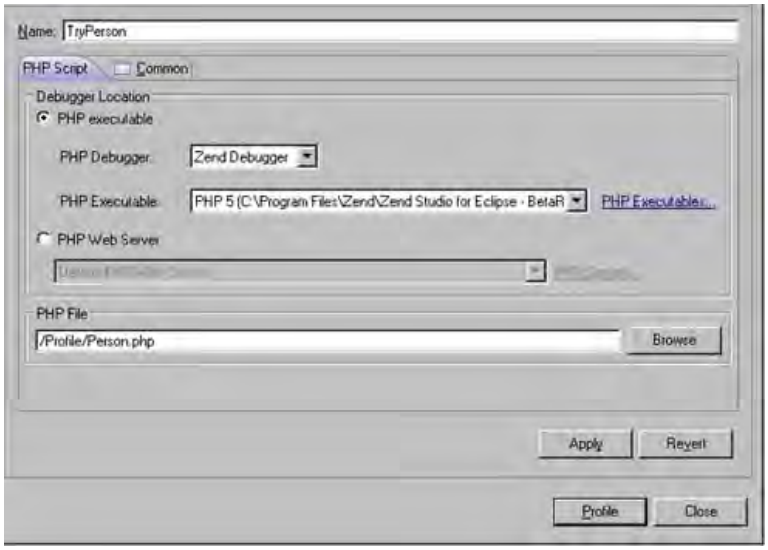
Remotely Profiling a PHP Script - Using your server's Zend Debugger.

Locally Profiling a PHP Script

This procedure describes how to Profile a PHP Script from your workspace using Zend Studio's internal debugger:

To locally Profile a PHP Script:

1. Click the arrow next to the Profile button on the toolbar and select Open Profile Dialog -or- from the main menu go to Run | Open Profile Dialog -or-right-click in PHP Explorer view and select Open Profile Dialog.
2. A Profile dialog will appear.



3. Double-click the PHP Script option to create a new Profile configuration.
4. Enter a name for the new configuration.
5. Ensure that the PHP Executable setting is selected under the Debugger Location category.
6. Select the required PHP executable.
7. Under PHP File, click Browse and select the required file.
8. Click Apply and then Profile.
9. A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.
   Click Yes. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with various Profiling views.

See the PHP Profile Perspective topic for more on the information displayed once a Profile session has been run.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Remotely Profiling a PHP Script

This procedure describes how to profile a PHP Script from your workspace using the Zend Debugger installed on your remote server.

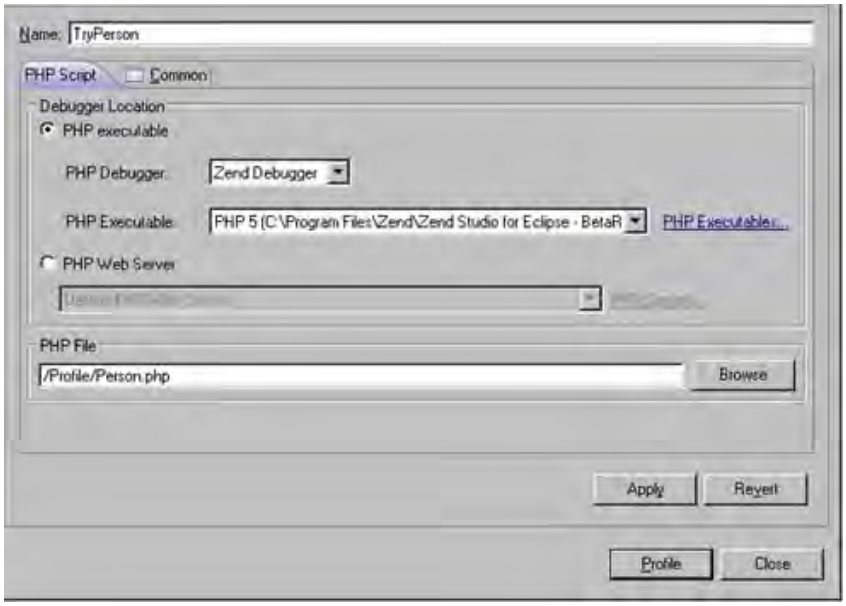
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To remotely profile a PHP script:

1. Click the arrow next to the Profile button on the toolbar and select Open Profile Dialog -or- from the main menu go to Run | Open Profile Dialog -or- right-click in PHP Explorer view and select Open Profile Dialog.
2. A Profile dialog will appear.



3. Double-click the PHP Script option to create a new Profile configuration.
4. Enter a name for the new configuration.
5. Select the PHP Web Server option and select your server from the drop-down list. If you have not yet configured your server, click the underlined 'PHP Servers' shortcut. The Servers preferences page will open.

Configure your server by following the instructions on adding a new server under the PHP Servers Preferences page.

For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.

6. Under PHP File, click Browse and select the required file.
7. Click Apply and then Profile.
8. A confirmation dialog will be displayed asking whether you want to open the Profiling Perspective.

   Click Yes. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with various Profiling views.

See the PHP Profile Perspective for more on the information that will be displayed once a profile session has been run.

Note:

If the file contains 'include' or 'require' calls to files which are not contained within the project, you must add them to the project's Include Path in order to simulate your production environment.

Profiling a PHP Web Page

This procedure describes how to profile whole applications, projects, files or collections of files that are already on the server.

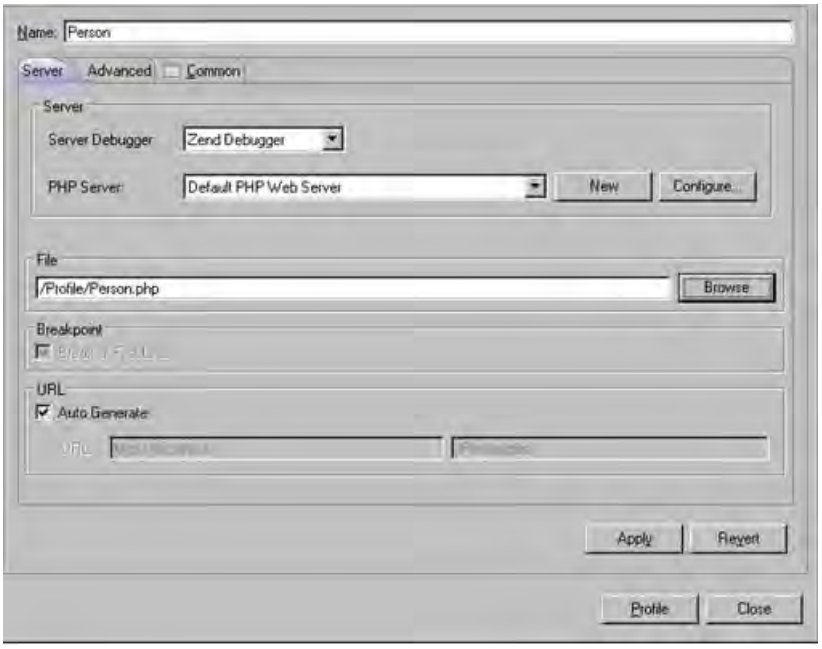
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To profile a PHP Web Page:

1. Click the arrow next to the Profile button on the toolbar and select Open Profile Dialog -or- go to Run | Open Profile Dialog from the main menu -or- right-click in PHP Explorer view and select Open Profile Dialog.
2. A Profile dialog will appear.
3. Double-click the PHP Web Page option to create a new Profile configuration.



4. Enter a name for the new configuration.
5. Select your server from the list.

   If you have not yet configured your server, click New. The PHP Server Creation

Wizard will open.

Configure your server by following the instructions on 'adding a new server' under the PHP Servers Preferences page.

For more information on configuring the communication between Zend Studio and your remote server, see Setting Up Remote Debugging.

6. Under PHP File, click Browse and select your 'debug target' file (the file from which the profiling process will start.)
7. For further profiling options, select the Advanced tab, which has the following options:
   - Open in Browser - Mark if you would like the application to be displayed in Zend Studio's internal browser
   - Source Location - Choose whether the source files used during this session will be taken from the Server or from a local copy (if a local copy is not available, files will be taken from the server according to the search mechanism detailed in the Path Mapping topic).
8. Click Apply and then Profile.
9. Click Yes if asked whether to open the PHP Profile Perspective. (If you would like the Profiling Perspective to open by default in the future, mark the 'Remember my decision' checkbox.)

The Profiling Perspective will open, displaying the Profiling Monitor window with various Profiling views.

See the PHP Profile Perspective for more on the information that will be displayed once a profile session has been run.

Profiling a URL

This procedure describes how to profile a URL.

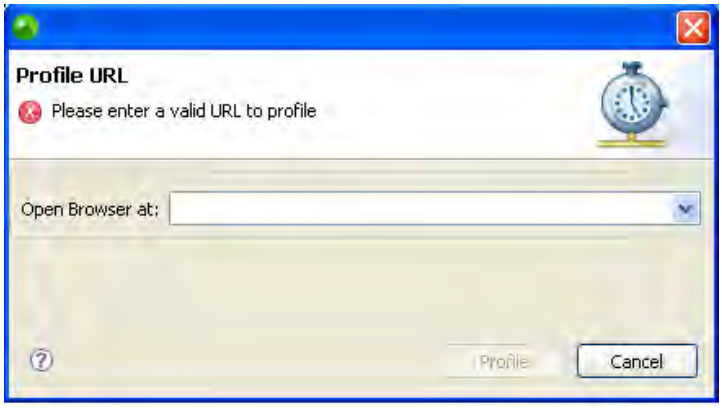
Note:

Your server must be running the Zend Debugger or XDebug in order for remote debugging and profiling capabilities to function.

The Zend Debugger comes bundled with Zend Server, Zend Core and Zend Platform, but can also be downloaded as a separate component from http://www.zend.com/en/products/studio/downloads.

To Profile a URL:

1. Click the profile URL button on the main toolbar -or- go to Run | Profile URL.
2. The Profile URL dialog will appear.



3. In the 'Open Browser at:' field, enter the URL of the page that should be profiled.
4. Click Profile.

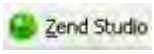
The Profile Perspective will open with a number of views detailing information about the profiling process.

See the PHP Profile Perspective for more on the information that will be displayed once a profile session has been run.

Profiling Using the Zend Browser Toolbar

This procedure describes how to Profile using the Zend Browser Toolbar.

To Profile using the Zend Browser Toolbar:

1. Ensure the Zend Browser Toolbar is installed on your browser.
2. Open your browser and browse to the page from which you would like to start profiling.
3. Ensure Zend Studio is open. You can open it by clicking the Zend Studio button 

in the toolbar.
4. Click the

button on the Zend Browser Toolbar to profile the page currently displayed in the browser.

The relevant Profile session will be launched in Zend Studio.

If Zend Studio is not open, you will be prompted to open it before the profiling session is launched.

Managing PHP Libraries

PHP Libraries Preferences

The PHP Libraries Preferences page allows you to create and maintain an external code library. Enabling PHP Libraries in your project allows libraries to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

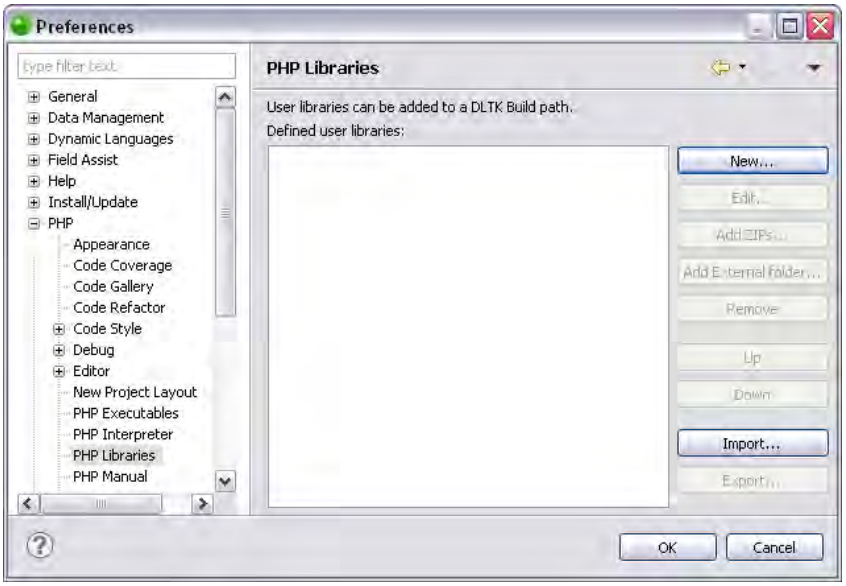
Note:

You may change the order your user libraries are in by using the **Up** and **Down** buttons. The order in which the libraries are arranged in this page defines the order they are available in Zend Studio's functionality, such as content assist.

The PHP Libraries Preferences page allows you to do the following:

- Add PHP Libraries
- Add External Folders to PHP Libraries
- Export PHP User Libraries
- Import PHP User Libraries
- Edit PHP User Libraries
- Edit PHP Library Components or Folders
- Remove a PHP Library or Library Folder

The PHP Libraries Preferences page is accessed from **Window** | **Preferences** | **PHP** | **PHP Libraries**.



Note:

Once you have added a user library in the PHP preferences page, you must also add it to your PHP Include Path of the project in which you would like to have it available. For more information see Configuring a Project's PHP Include Path.

Adding a PHP Library

This procedure describes how to add a user library to Zend Studio. Including user libraries in your project or environment saves you time in writing and debugging code, as you are re-using debugged code.

To add an additional PHP Library to your project:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **New**.
   The "New User Library" Dialog will open.
3. In the "New User Library" dialog, enter the name of your user library
4. Select the "Add to environment" checkbox if you would like this library to be added to your entire environment instead of a specific project.
5. To apply changes click **OK**.

A new empty library will be added to the list. Next to the name in brackets indicates if it is shared by the environment or only related to a project.

Adding a PHP library creates a place folder in which you can place external files that contain pre-written code. For more information see Adding External Folders to PHP Libraries.

Note:

Once you have added a user library in the PHP preferences page, you must also add it to your PHP Include Path of the project in which you would like to have it available. For more information see Configuring a Project's PHP Include Path.

Adding External Folders to PHP Libraries

This procedure describes how to add external folders to a user library. This allows you to compile a user library using folders from varying places on your computer. In order to add an external folder, you must have already created a user library. For more information see Adding a PHP Library.

If you are interested in using compressed files, click the **Add ZIPs....** button and select a file instead of following the procedure below.

To add an external folder to your user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **Add External folder...**
   The "External folder Selection" dialog will open.
3. Select a folder in the "External folder Selection" dialog or create a new folder by clicking **Make New Folder**.
4. To apply changes click **OK**.

The folder will be added to the PHP library.

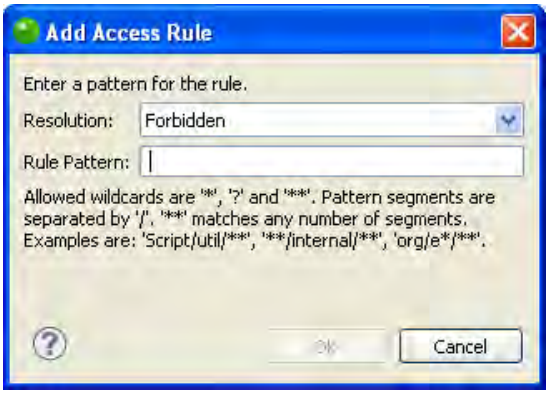
The folders that are included in the user defined libraries allow you to control their access rules. The access rule default setting is "No restrictions." See Defining Access Rules below for more information.

Defining Access Rules

This procedure describes how to define access rules. Access rules allow you to control where your libraries will be available. From here you can control what code completion options will be available in features such as Content Assist on a project level.
In order to define access rules you must have an external PHP folder added to your project. For more information Adding External Folders to PHP Libraries.

To define an access rule for your user defined library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page, double click "Access rules:" located in the hierarchal tree below your added folder. The default setting is "No restrictions."
   The "Type Access Rules" dialog will open.
3. Click **Add...**
   The "Add Access Rule" dialog will open.


Add Access Rule
Enter a pattern for the rule.
Resolution:
Forbidden
Rule Pattern:
Allowed wildcards are '*', '?' and '**'. Pattern segments are separated by '/'. '**' matches any number of segments.
Examples are: 'Script/util/**', '**/internal/**', 'org/e*/**'.
Cancel

4. From the "Resolution:" drop down menu select one of the following:

- Forbidden - This will make the file completely unaccessible in the relevant project.
- Discouraged - This will make the file available but not highly suggested.
- Accessible - This will allow the project to access the file and its contents freely.
- In the "Rule Pattern" field, insert a rule pattern to define a pattern for the rule. Use wildcards to create a pattern. The accepted wildcard values are:
  - '*'
  - '**'
  - '?'

This will automatically apply the rule to any of the files that match the pattern.

To apply changes click **OK**.

5. In the "Type Access Rules" dialog, you also have the option to edit, remove, or change the order of your defined access rules by clicking the buttons. Before a file is allowed access into your project, it will first confirm that none of the access rules apply to it. These rules are checked in the order they are placed here.

Your PHP Library now has an access rule defined in the hierarchal tree in the PHP Libraries Preferences page. You may edit a file's access rule at any time by repeating this procedure.

Importing PHP User Libraries

This procedure describes how to import existing user libraries that are on the disk or in a repository. This allows you to take an already built library and use it in your project, as well as share a library with other users of the same repository. Importing a library will only import a description of the library in .xml format, and will not include any of the library's content. Before importing a user library, you must first have access to an exported user library. For more information see Exporting User Libraries.

To import a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page click **Import...**
   The "Import User Libraries" dialog will open.
3. To choose where you would like to import your library from, fill in the "File location:" text field with the URL or click **Browse...** and select the location.
4. Select the libraries you would like to import from the options in the "Libraries contained in the selected file:" box, or press **Select All** or **Deselect All**.
5. To apply changes click **OK**.
   Your library's description in .xml format has now been imported into Zend Studio.

If the library of the user who imports it is stored in the same location on the disk as the user who exported it, Zend Studio will automatically find the libraries content and store it accordingly. If the user library is stored in a different place for the two users, the **Edit...** button allows you to replace the location URL. See Editing PHP User Libraries for more information.

Exporting PHP User Libraries

This procedure describes how to export user libraries, making them accessible to whoever has access to the repository where it is stored. Exporting a user library will only export a description of the library in .xml format, and will not include any of the library's content.

To export a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library you would like to export and click **Export...**
   The "Export User Libraries" dialog will open.
3. Select the library you would like to export by clicking the check box beside it. You may also use the **Select All** or the **Deselect All** buttons.
4. To choose where you would like to export your library to, fill in the "File location:" text field with the URL, or click **Browse...** and select the location.
5. To apply changes click **OK**.

Your user library has now been exported to the location you specified.
You may now import the exported user libraries from any location that has access to the location in which it is stored. For more information see Importing PHP User Libraries.

Editing PHP Library Components or Folders

This procedure describes how to edit user library folders. Edit a user library folder when you have imported a folder who's original location is different from its content's location on your disk. For more information see Importing PHP User Libraries.

To edit a library component:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library folder to edit and click **Edit...**

   The "Edit External folder" dialog will open.
3. Select the folder that contains the user library content, or create a new folder by clicking **Make a New Folder**.
4. To apply changes click **OK**.

You have now edited your user library's URL which will import the content from the proper location on the disk.

Note:

If the imported location of the selected file you are editing is different than the new location, the new location will overwrite it.

Editing PHP User Libraries

This procedure describes how to edit user libraries. You will need to edit a user library when you would like to change it's name. Changing a library's name allows you to create descriptive differentiations between your libraries. In order to edit a library, there must be libraries available. For more information see Importing User Libraries or Adding a PHP Library.

To edit a user library:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library you would like to edit and click **Edit...**
   The "Edit User Library" dialog will open.
3. Enter the user library name in the "User Library Name:" text field.
4. In order to add the given user library to your environment you must select the "Add to environment" box. Otherwise your library will be added on a project level.
5. To apply changes click **OK**.

You have now edited your user library. To edit a folder within your library see Editing Library Components.

Removing a PHP Library or Library Folder

This procedure describes how to remove a user library or library folder from Zend Studio.Removing a user library or library folder means that its contents will no longer be available in Zend Studio, including in it's functionality such as Content Assist and Refactoring.

Important Note:

If your PHP user library or library folder is associated with specific projects, removing it will delete it from the projects as well.

To remove a PHP library/library folder:

1. Go to **Window** | **Preferences** | **PHP** | **PHP Libraries**.
2. In the PHP Libraries Preferences page select the user library or library folder to remove and click **Remove**.(Removing a user library will remove all of the folders within it as well.)
   Your user library or library folder is deleted.
3. To apply changes click **OK**.

Your user library or library folder has been removed. If you would like to import a library see Importing PHP User Libraries. To add a folder to an existing library see Adding External Folders to PHP Libraries.

Configuring a Project's PHP Include Path

Adding external resources to a project's include path makes resources outside the project available for operations such as debugging, refactoring and content assist. See PHP Include Paths for more information.

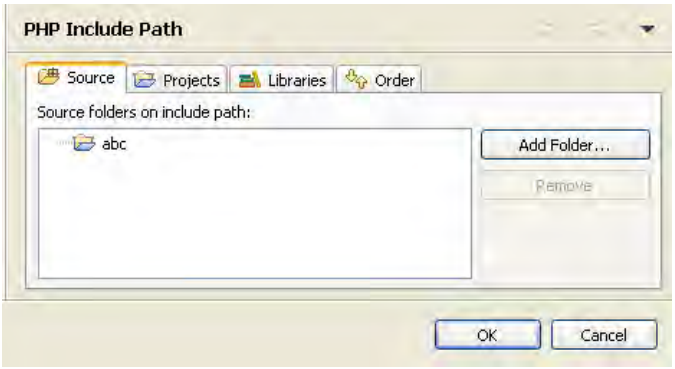
This procedure describes how to configure your project's Include Path.

To configure your project's Include Path:

1. In PHP Explorer view, right-click the required project and select **Include Path | Configure Include Path...**

   -Or- right-click the project and select **Properties | Include Path**.

   The project's PHP Include Path properties page will appear.



From this page you can configure the following:

To configure source folders on your Include Path:

i. Select the Source tab.

   By default, all folders which are on your Include Path will be added to your Build Path.

> **Note:**
> If your project root is on your Include Path, all folders and resources contained within your project will also be on the Include Path. To add only certain folders, remove the project root from the Include Path and add only the required folders.

ii. To remove a folder, select it and click **Remove**.

iii. To add a folder from your project, click the **Add Folder** button and select the required folder.

   A prompt will display, asking whether you would also like to add the selected folder(s) to your Build Path.

   It is recommended that the resources on your Build Path match the resources on your Include Path.

Click **Yes** to add the folder(s) to your Build Path or **No** for your Build Path to not be affected.

To add another project from your workspace to your Include Path:

i. Select the "Projects tab".

ii. Click **Add**.

The Required Project Selection dialog appears.

ii. Select the projects you would like to add and click **OK**.

The selected project(s) will be added to your project's Include Path.

To add a library to your Include Path:

i. Select the "Libraries tab".

ii. Click **Add Library**.

The Add Library dialog appears.

ii. Select the required Library (if available) and click **OK**.

To add external folders to your Include Path:

i. Select the "Libraries tab".

ii. Click **Add** External Folder.

The Add Include Path dialog appears.

ii. Browse to and select the required folder.

v. Click **OK**.

The folder will be added to your project's Include Path.

Note:
The library is a read-only file and will not be available for editing.

To configure the order of elements on your Include Path:

i. Select the "Order tab". This determines the order in which resources will be searched for in require/include calls.

See Include Paths for more information on the order for which files are searched for.

If applicable, it is recommended that elements appear in the same order as they do in your php.ini.

ii. If necessary, rearrange the order of the entries. To do so, select an element and click **Up** or **Down** to move it in the list.

2. Once all the elements are added and are in the right order, click **OK**.

All the selected elements will be added to the project's include path.

Configuring a Project's PHP Build Path

By configuring your project's PHP Build Path, you can select resources which will be included or excluded from the Build process.

Note:

If during project creation you selected the 'use project as source folder' under the Project Layout category, all resources within the folder will be added to the Build Path by default.

If you selected to 'create separate folders for source files and public resources' during the project's creation, resources in the 'public' folder will be excluded from the Build Path by default.

The project's PHP Build Path can be configured through the project's Build Path properties page, accessed by right-clicking the required project and selecting Build Path | Configure Build Path.

This page allows you to add folders and resources to your Build Path and configure rules for including / excluding certain resources in the project.

You can configure your Build Path according to the options listed below.

Once you have made changes, click OK to save.

Note:

By default, all folders which are on your Build Path will be included in your Include Path.

Configuring Inclusion/Exclusion Patterns for the Project

By configuring inclusion/exclusion patterns for the Build Path, you can select to include or exclude all resources in your project which match a defined pattern.

To include/exclude resources from the Build Path:

1. In the Build Path Properties page, expand the node next to the folder whose Inclusion/Exclusion pattern you want to configure and click Edit.
   -Or- In PHP Explorer view, right-click the folder whose Inclusion/Exclusion pattern you want to configure and select Build Path | Configure Inclusion / Exclusion Filters.
   The Inclusion and Exclusion patterns dialog is displayed.
2. Click Add next to the Inclusion or Exclusion patterns panes.
   The Add Inclusion/Exclusion Pattern dialog is displayed.
3. Enter or select the required resource(s) or pattern to include / exclude and click OK.
4. Click Finish.

All resources in the project which match an inclusion pattern but do not match an exclusion pattern will be added to the Build Path.

Configuring Different Inclusion/Exclusion Patterns for Folders Within your Project

You can configure different inclusion/exclusion rules for child folders within a selected directory. However, you will need to exclude the selected child folder from the parent directory's Build Path and add it as a separate Build Path source folder. This is done to resolve any conflicts which may arise from configuring one pattern for the parent folder and another for the child.
Source folders can be created and/or added from the PHP Build Path properties page or from PHP Explorer view.

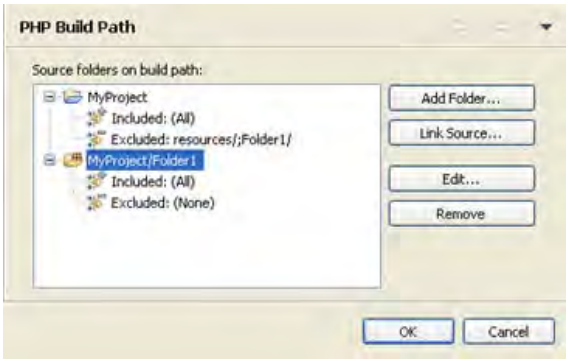
To add a directory as a separate source folder:

From the PHP Build Path Properties page:

1. Click Add Folder...
   The Source Folder Selection dialog will display.
2. If necessary, click Create New Folder to create a new folder.
3. Select the required folder and click OK.
4. The folder will be added as a separate source folder in the Build Path list.

Note:

In order to overcome the nesting conflict, you should exclude the folder you have just added from the parent folder's Build Path. See 'Configuring Inclusion/Exclusion Patterns', above, for information on how to exclude the folder and all resources contained within it.



From PHP Explorer view:

1. Right-click the project and select Build Path | New Source Folder.
   The New Source Folder dialog will be displayed.

New Source Folder
Source folder
Add a new source folder relative to '/abc'.
Folder name:
Replace existing project source folder entry to solve nesting
Update exclusion filters in other source folders to solve nesting
< Back
Next >
Finish
Cancel

2. Enter the name for the new folder in the Folder name field.
3. Adding a folder to a project's Build Path could cause a conflict between the inclusion/exclusion patterns configured for the project root and those configured for the selected folder.

   In order to avoid these nesting conflicts, select one of the following options:

   - Replace existing project source folder entry to solve nesting - This will remove your parent directory as a source folder from the Build Path configuration and replace it with the currently selected folder.
   - Update exclusion filters in other source folders to solve nesting - This will exclude the currently selected folder from the parent directory's Build Path but will add it as a separate entity so that different inclusion/exclusion patters can be configured for it.
4. Click Finish.

You can now configure a pattern for including/excluding resources for the source folder by following the instructions under 'Configuring Inclusion/Exclusion Patterns', above.

Adding External Source Folders to the Build Path

You can add an external source folder as a link to your project which will be scanned during the Build process.

To add an external source folder to the project's Build Path:

1. In the Build Path Properties page, click the 'Link Source...' button.
   -Or- In PHP Explorer view, right-click the project and select Build Path | Link Source.
   The Link Source dialog is displayed.

2. Click the 'Browse..' button and browse to the location of the folder you want to add to the Build Path.
   Alternately, click Variables and select the variable which points to the required resource.
3. The Folder name field will have been automatically populated with the name of the original folder.
   Edit this entry if required.
4. Adding a folder to a project's Build Path could cause a conflict between the inclusion/exclusion patterns configured for the project root and those configured for the selected folder.
   In order to avoid these nesting conflicts, select one of the following options:
   - Replace existing project source folder entry to solve nesting - This will remove your parent directory as a source folder from the Build Path configuration and replace it with the currently selected folder.
   - Update exclusion filters in other source folders to solve nesting - This will exclude the currently selected folder from the parent directory's Build Path but will add it as a separate entity so that different inclusion/exclusion patters can be configured for it.

- Ignore nesting conflicts - The selected folder will be added as a separate entity to the Build Path list but will need to be manually excluded from the project root's Build Path to avoid nesting conflicts.

5. Click Finish.

The folder will be added to your Build Path and displayed in the Build Path list.
You can now configure a pattern for including/excluding resources for the linked folder by following the instructions under 'Configuring Inclusion/Exclusion Patterns', above.

Adding a Server Location Path Map

This procedure describes how to add a Path Map to a server so that files which are called from a certain location on the server will be searched for in a local location during remote PHP Script debbuging/profiling and PHP Web Page debugging/profiling. This will only apply when the 'use local copy' option is selected in the Advanced tab of the PHP Web Page debugging configuration).

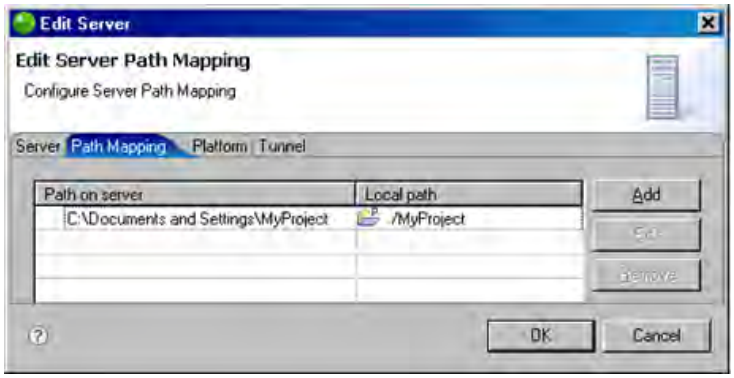
See Path Mapping for more details

To add a Path Map to a server:

1. Open the PHP Servers Preferences Page by going to Window | Preferences on the Menu Bar and selecting PHP | PHP Servers from the Preferences list.
2. Select the server on which you would like to create the Path Map and click Edit.
3. In the Edit Server dialog, select the Path Mapping tab.
4. Click Add.
5. An Add new Path Mapping dialog appears.
6. Enter the Server Path from which you would like to create the Path Map. Files called from this location will be searched for in the path specified below.
7. Select either the 'Path in Workspace' or 'Path in File System' option and click Browse to specify the location.

8. Click OK.

Your Path Map will be added to your server list.



The next time a file is called from the Path on Server, it will be searched for in the local location you have specified.

Note:

Path Mapping can also be set automatically during Debugging / Profiling. See the Path Mapping topic for more details.

Using PHPUnit Testing

The following tasks will guide you through the process of creating, running and reporting on PHPUnit Test Cases and Suites:

- Creating a PHPUnit Test Case
- Running a PHPUnit Test Case
- Creating a PHPUnit Test Suite
- Running a PHPUnit Test Suite
- Reporting on PHPUnit Test Results

Creating a PHPUnit Test Case

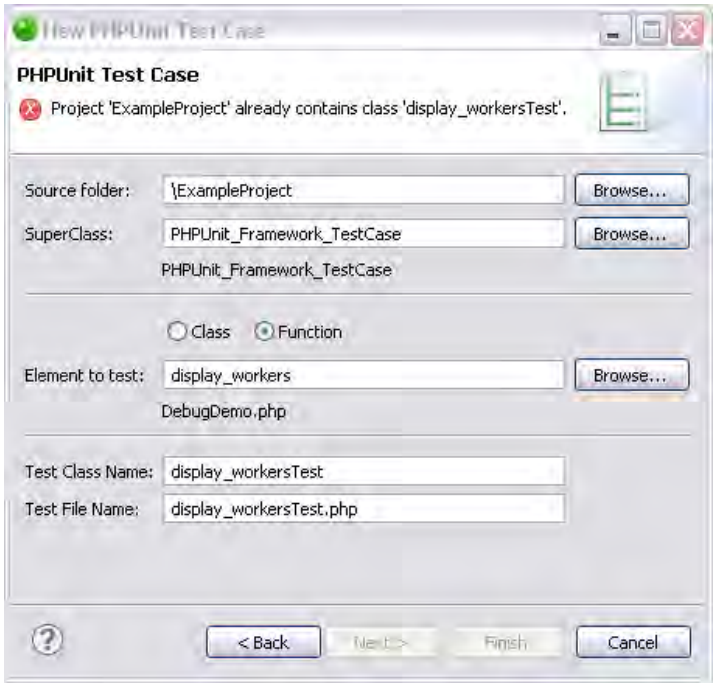
This procedure describes how to create a PHP Unit test case.

Zend Studio will automatically create test case files which can be run in order to check the functionality of your code.

You must first create a file containing a class with functions which will be tested when the PHPUnit Test case is run.

To create a PHPUnit Test Case:

1. In PHP Explorer view, right-click the file containing the classes you would like to test and select New | Other | PHP | PHPUnit | PHPUnit Test Case.
   The PHPUnit Test Case dialog will open, with relevant information already entered into the various fields.
   Note that a new file will be created called "FileName"Test.php



2. A SuperClass is a class from which the new PHPUnit Test Case will inherit functionality (e.g. setup and constructors). If necessary, click Browse next to the 'SuperClass' field to select a different PHPUnit Framework SuperClass.
3. Click Browse next to the 'Element to test' field to select the Class or Function which will be tested in the new PHPUnit Test Case.
4. If this is the first PHPUnit Test created for the project, a warning will appear stating that the PHPUnit is not on the include path of your project.

To add it to the include path, click the underlined "Click here" link at the bottom of the dialog screen. This will enable PHPUnit Content Assist options in the PHPUnit Test.

Once it has been clicked, the link and the warning message will disappear.

5. Click Finish to create your test case.
6. The new test file, containing tests for the selected elements, will be added to your project.

   Note that all relevant functions in the original class will have a corresponding test function in the test file.

   However, test functions will have been created with no parameters.
7. Before you can run your test file, you must create relevant tests and parameters for each of your functions, depending on the results you expect to see when the function is run. For each function, write a test with demo input parameters and the expected result. When the test is run, Zend Studio will insert these parameters into your original file's functions to see if the result is as expected.

Once you have completed the file by creating relevant test functions and inserting parameters, your PHPUnit test case is ready to be run.

Running a PHPUnit Test Case

This procedure describes how to run a PHPUnit Test Case and how to analyze the results.

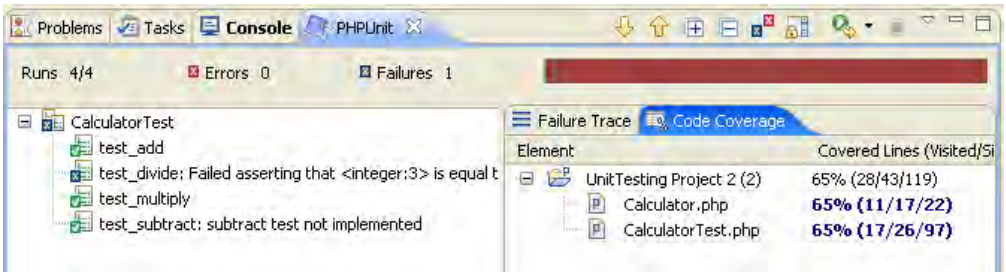
Before running a PHPUnit Test Case, one needs to be created by following the instructions under 'Creating a PHPUnit Test Case'.

To Run a PHPUnit Test Case:

1. Open your PHPUnit Test Case file in the editor.
2. To run the PHPUnit Test Case, click the arrow next to the Run on the toolbar and select Run As | PHP Unit Test -or- from the Main Menu, go to Run and select Run As | PHP Unit Test -or- right-click the file in PHP Explorer view and select Run As | PHP Unit Test

   -Or- to debug the PPHUnit Test Case, click the arrow next to the debug button on the toolbar and select Debug As | PHP Unit Test -or- from the Main Menu, go to Run and select Debug As | PHP Unit Test -or- right-click the file in PHP Explorer view and select Debug As | PHP Unit Test

   The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.



3. In the main area of the PHPUnit test view, the results for each of the tests run will be displayed.

   Tests that have passed successfully will be displayed with a green tick icon.

   Tests that have failed will be displayed with a blue X icon.

   Functions with tests that have not been implemented (i.e. functions that tests have not been created for), will have passed but will have a note indicating that they have not been implemented.
4. The number at the top of the view indicates how many tests have been run. Tests may not be run if an 'exit' command is given or if a fatal error is encountered.
5. Click the 'Show failures only' icon to only view failed results.

6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon to display only functions relevant to your application and not PHPUnit functions.
7. Double-click on a failed result to be taken to the test function in the test file.
   To correct the failed result, either fix the test function or the original function on which it was run.
8. The Code Coverage display indicates how much of the code in both the original file and the test file was run:
   - The percentage in the Covered Lines column displays the percentage of lines executed out of the total number of executable lines.
   - The number of 'visited' lines are the number of executable code lines.
   - The number of 'significant' lines are the number of significant (i.e. executable) lines.
   - The number of 'total' lines is the total number of lines in the file.
9. Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run.
   'Visited' lines will be highlighted in blue.
   'Significant' lines will be highlighted in pink.

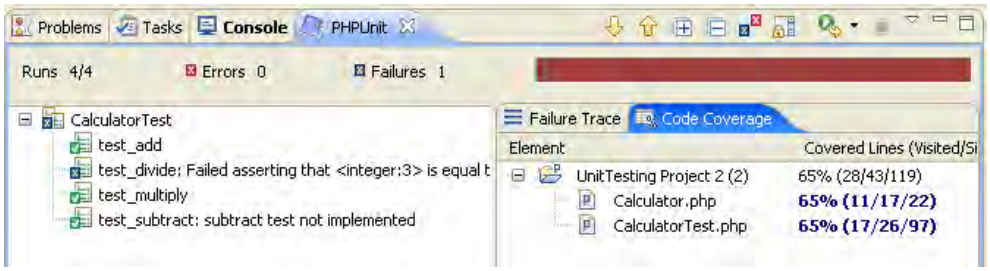
Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button in the PHPUnit view until all tests pass successfully.

To Debug a PHPUnit Test Case:

1. Open your PHPUnit Test Case file in the editor.
2. Click the arrow next to the Run button on the toolbar and select Run As | PHP Unit Test &endash;or- from the Main Menu, go to Run and select Run As | PHP Unit Test .

   -Or- to debug the PPHUnit Test Case, click the arrow next to the debug button on the toolbar and select Debug As | PHP Unit Test &endash;or- from the Main Menu, go to Run and select Debug As | PHP Unit Test .

   The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.



3. In the main area of the PHPUnit test view, the results for each of the tests run will be displayed.

   Tests that have passed successfully will be displayed with a green tick icon.

   Tests that have failed will be displayed with a blue X icon.

   Functions with tests that have not been implemented (i.e. functions that tests have not been created for), will have passed but will have a note indicating that they have not been implemented.
4. The number at the top of the view indicates how many tests have been run. Tests may not be run if an 'exit' command is given or if a fatal error is encountered.
5. Click the 'Show failures only' icon to only view failed results.
6. Select a failed result to view it in the Failure Trace view. Click the Filter Stack Trace icon to display only functions relevant to your application and not PHPUnit functions.
7. Double-click on a failed result to be taken to the test function in the test file.

   To correct the failed result, either fix the test function or the original function on which it was run.

8. The Code Coverage display indicates how much of the code in both the original file and the test file was run:
   - The percentage in the Covered Lines column displays the percentage of lines executed out of the total number of executable lines.
   - The number of 'visited' lines are the number of executable code lines.
   - The number of 'significant' lines are the number of lines which were executed.
   - The number of 'total' lines is the total number of lines in the file.
9. Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run.
   'Visited' lines will be highlighted in blue.
   'Significant' lines will be highlighted in pink.

Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button in the PHPUnit view until all tests pass successfully.

You can transform the xml file created with while running the PHPUnit Teasr into an HTML report using the Report Generator icon . See Reporting on PHPUnit Test Results for more details.

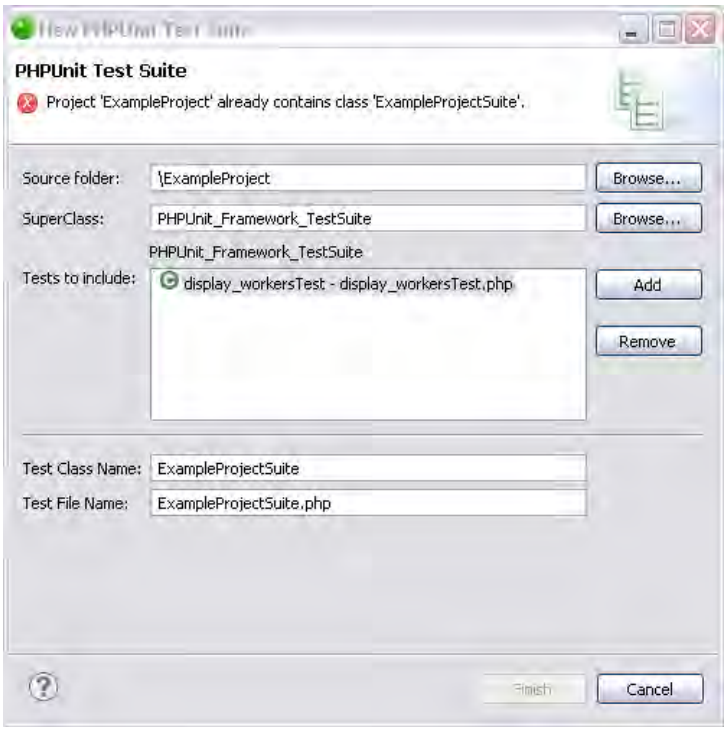
Creating a PHPUnit Test Suite

This procedure demonstrates how to create a PHPUnit Test Suite for running a number of PHPUnit Test Cases at once. This function is useful if you have a number of tests which you would like to unify into one.

Before creating the PHPUnit Test Suite, you must have created all your separate PHPUnit Test Cases.

To create a PHPUnit Test Suite:

1. In PHP Explorer View, right-click the project which contains your PHPUnit Test Cases and select New | Other | PHP | PHPUnit | PHPUnit Test Suite.
   The New PHPUnit Test Suite dialog appears.



2. The 'Tests to include' category will show the available test cases within the project. Cilck Add to choose the test cases you would like to include in the Test Suite.
3. Click Finish.

A PHPUnit Test Suite will be created, integrating all the separate PHPUnit test cases, and will be added as a file to your project.

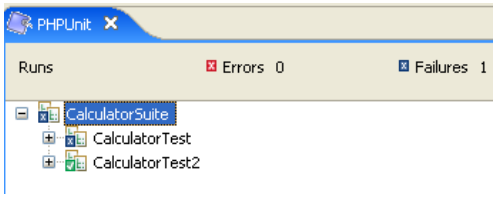
Running a PHPUnit Test Suite

This procedure describes how to run a PHPUnit Test Suite and how to analyze the results.

Before running a PHPUnit Test Suite, one needs to be created by following the instructions under 'Creating a PHPUnit Test Suite'.

To run a PHPUnit Test Suite:

1. Open your PHPUnit Test Suite.
2. Click the arrow next to the Run button on the toolbar and select Run As | PHP Unit Test -or- from the Main Menu, go to Run and select Run As | PHP Unit Test -or- right-click the file in PHP Explorer view and select Run As | PHP Unit Test.
   All the PHPUnit Test Cases contained inside the PHPUnit Test Suite will be run.



3. The PHPUnit view will be displayed, with a section showing all the tests run and the results, and two extra tabbed views showing code coverage and failure trace.
   The results of the individual PHPUnit Test Cases will be displayed in a tree diagram.
4. Expand the nodes to see the results for each of the individual test cases.
   Tests that have passed successfully will be displayed with a green tick icon.
   Tests that have failed will be displayed with a blue X icon.
   Tests that have not been implemented (i.e. that tests have not been written for), will have passed but will have a note indicating that they have not been implemented.
5. Double-click on a failed result (if applicable) to be taken to the test function in the test file.
   To correct the failed result, either fix the test function or the original function on which it was run.
6. The Code Coverage display indicates how much of the code in both the original file and the test file was run.
   Click on the code coverage statistics next to each file to open the Code Coverage view displaying the code with the lines of code that were run highlighted in blue.
7. Once you have corrected errors, you can re-run the PHPUnit Test by clicking the Run Last Test button in the PHPUnit view until all tests pass successfully.

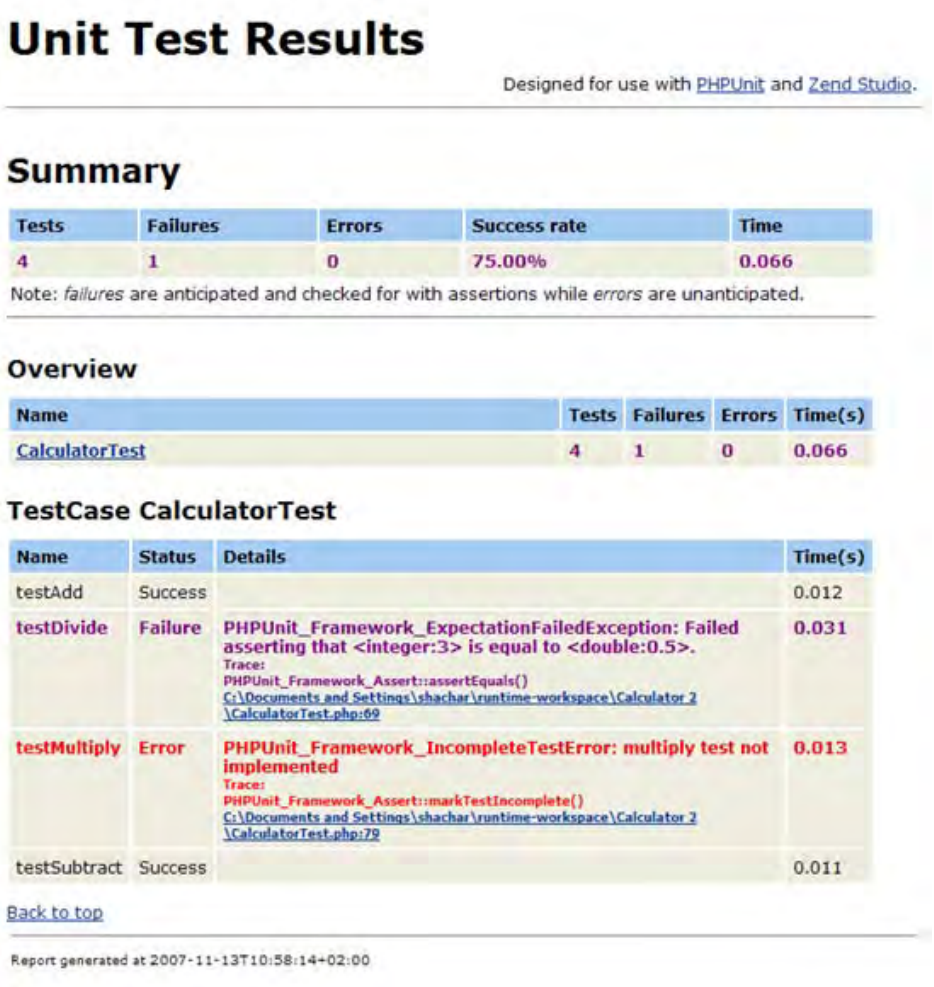
Reporting on PHPUnit Test Results

Once you have run a PHPUnit Test Case/Suite, you can quickly and easily create a report to view the results of your test.

To generate a report:

1. Run a PHPUnit Test Case/Suite. (See Running a PHPUnit Test Case or Running a PHPUnit Test Suite for more information).
2. In the PHPUnit view, click the arrow next to the Report Generator icon on the view's toolbar to select a report type -or- click the Report Generator icon itself to generate the last generated report.
   See PHPUnit Testing for more on the different types of reports.

3. A report will be automatically generated and opened in a browser window.



Unit Test Results

Designed for use with PHPUnit and Zend Studio.

Summary

Tests	Failures	Errors	Success rate	Time
4	1	0	75.00%	0.066

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Overview

Name	Tests	Failures	Errors	Time(s)
CalculatorTest	4	1	0	0.066

TestCase CalculatorTest

Name	Status	Details	Time(s)
testAdd	Success		0.012
testDivide	Failure	PHPUnit_Framework_ExpectationFailedException: Failed asserting that <integer:3> is equal to <double:0.5>. Trace: PHPUnit_Framework_Assert::assertEquals() C:\Documents and Settings\shachar\runtime-workspace\Calculator 2 \CalculatorTest.php:69	0.031
testMultiply	Error	PHPUnit_Framework_IncompleteTestError: multiply test not implemented Trace: PHPUnit_Framework_Assert::markTestIncomplete() C:\Documents and Settings\shachar\runtime-workspace\Calculator 2 \CalculatorTest.php:79	0.013
testSubtract	Success		0.011

Back to top

Report generated at 2007-11-13T10:58:14+02:00

Note:

Reports will be generated in the location defined in the PHPUnit Preferences page.

4. Clicking the link beneath a failed test result will take you to the relevant test.

Using Refactoring

The Refactoring feature allows you to:

- Rename Files
- Rename Elements
- Move Files
- Extract Variables
- Extract Methods

Note:

Refactoring options will only be available from within PHP Explorer view and not from Navigator view.

Using the Navigator view's move/rename functions will not update any referenced instances of the file/element.

Renaming Files

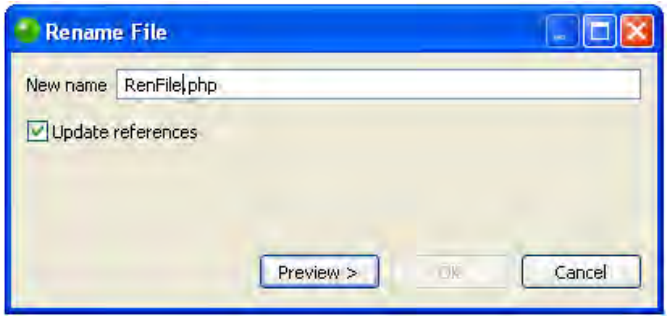
This procedure describes how to rename files and update all instances where that file is referenced within the project.

Note:

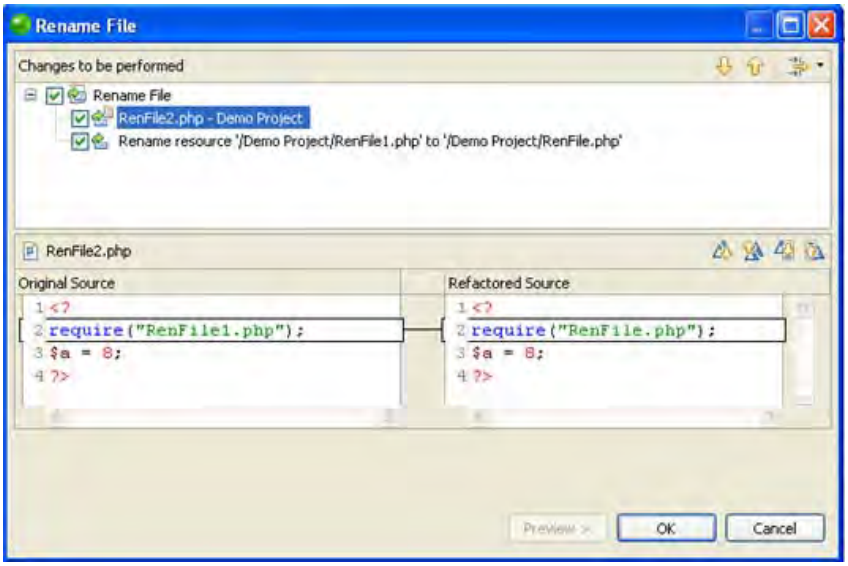
Ensure that you save any changes to the file before applying the refactoring feature to it.

To rename a file using the Refactoring feature:

1. In PHP Explorer view, right-click the file which you would like to rename and select Refactor | Rename -or- select it and go to Refactor | Rename from the Menu Bar.
   A Rename File dialog will appear.



2. Enter the file's new name.
3. Check the "Update references" box and click Preview.
   A preview window will open with a change tree showing all the changes which will be made to reflect the rename of the file, sorted according to the files in which the changes will be made.



Note that, if the file has been referenced (required, included etc.) in other files, the name of the file will also be updated in those instances.

4. You can scroll through the different changes using the Select Next / Previous Change scrolling arrows .
5. If you are satisfied with the changes, press OK.

The file will be renamed and all instances where that file is referenced will be updated to reflect the change.

Renaming Elements

This procedure describes how to rename a PHP element and ensure that all references to that element are updated.

All PHP elements can be renamed and refactored from PHP Explorer view. The following is a list of applicable PHP elements:

- Classes
- Interfaces
- Variables
- Methods
- Functions
- Constants
- Class Members

Note:

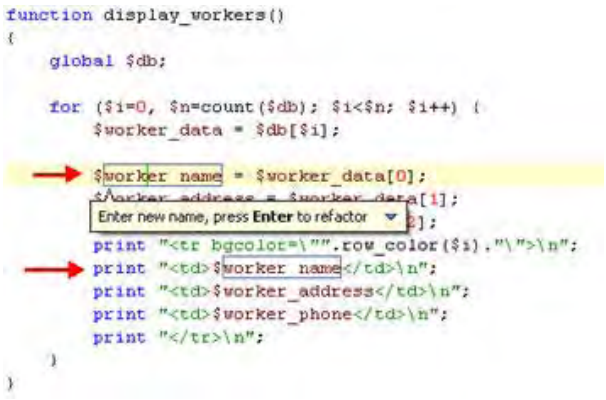
Ensure that you save any changes to the file before applying the refactoring feature.

Elements can either be renamed from within the editor itself if the In-place refactoring feature is activated (will not display a preview of changes) or from the Rename Element dialog.

Renaming Elements within the Editor

To rename an element within the editor:

1. Ensure the 'Rename in editor without dialog' checkbox is marked in the Code Refactor Preferences page (accessible from Window | Preferences | PHP | Code Refactor).
2. In the editor, place your cursor on the element to be renamed.
3. From the menu bar select Refactor | Rename -or- right-click and select Refactor | Rename -or- press Alt-Shift-R.
   All occurrences of the element are put in a frame and the Refactor popup is displayed.



4. Type the new element name in the box.
   All occurrences of the element name are automatically updated.

Note:
To preview the changes before applying them, click the arrow in the right-hand corner of the Refactor popup and select Preview.

5. Click Enter to apply the refactoring.
   A dialog is displayed prompting you to save the file before the refactoring is applied.
6. Mark the 'Always save all modified resources automatically prior to refactoring' checkbox so that the dialog will not be displayed again.
7. Click OK.

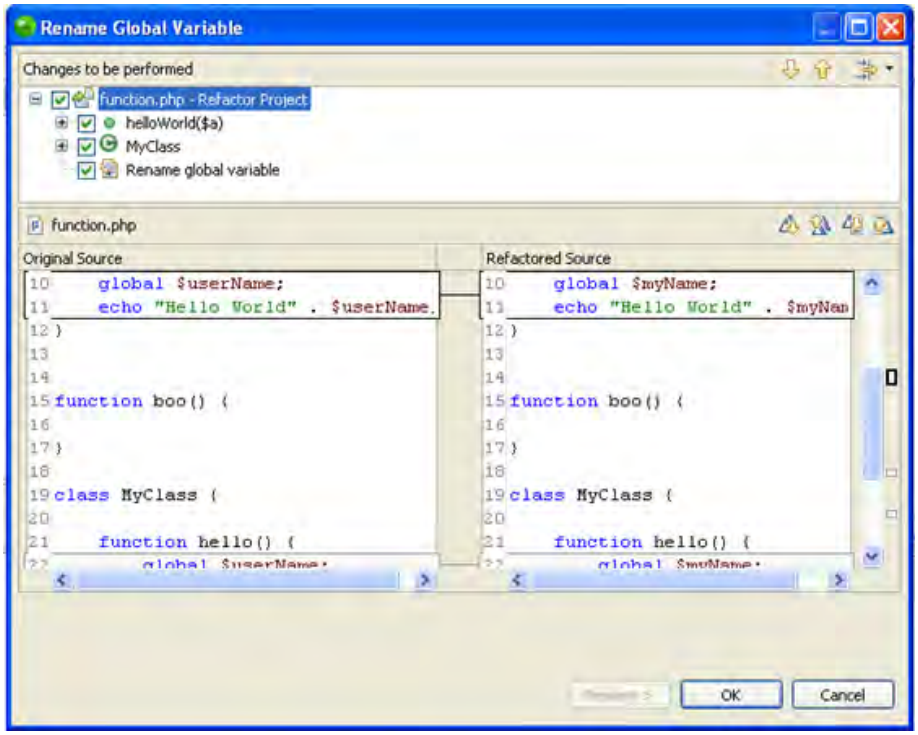
The element will be renamed and all instances where that element is referenced will be updated to reflect the changes.

Renaming Elements through the Refactor Dialog

To rename an element through the Refactor dialog:

1. Ensure the 'Rename in editor without dialog' checkbox is unmarked in the Code Refactor Preferences page (accessible from Window | Preferences | PHP | Code Refactor).
2. In the editor, place your cursor on the element to be renamed or select it in the PHP Explorer.
3. From the menu bar select Refactor | Rename -or- right-click and select Refactor | Rename -or- press Alt-Shift-R
   The Rename dialog box will be displayed. The name of the dialog will be dependent on the element type.

2. Enter the element's new name. You must enter a valid name for the required element - i.e. one that starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
3. Check the "Update textual occurrences in comments and strings" box if you want the element's name to be updated in all comments and strings where it is referenced. This will force you to preview the changes before applying them.
4. Click OK to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
5. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the rename of the element.
   The changes will be listed according to the context within which they appear. You can therefore expand the nodes to see all changes within particular files, classes or functions.
6. Use the Next / Previous Change arrows to scroll through all possible changes. Unmarking the checkboxes next to the changes will cause those changes not to take effect.



Note that if changes will be made in other files which reference the element being refactored, the changes will also be listed here under the file name.

7. The changes to be applied will be displayed in the bottom pane.
   You can scroll through the different changes using the scrolling arrows:
   - Next / Previous Difference scrolling arrows - Scroll through changes to be applied within the element selected in the top pane.
   - Next / Previous Change scrolling arrows - Scroll through all changes to be applied. If you unmarked changes in the top pane, these will not be displayed when using these arrows.
8. Once you are satisfied with the changes, click OK.

The element will be renamed and all instances where that element is referenced will be updated to reflect the changes.

Moving Files

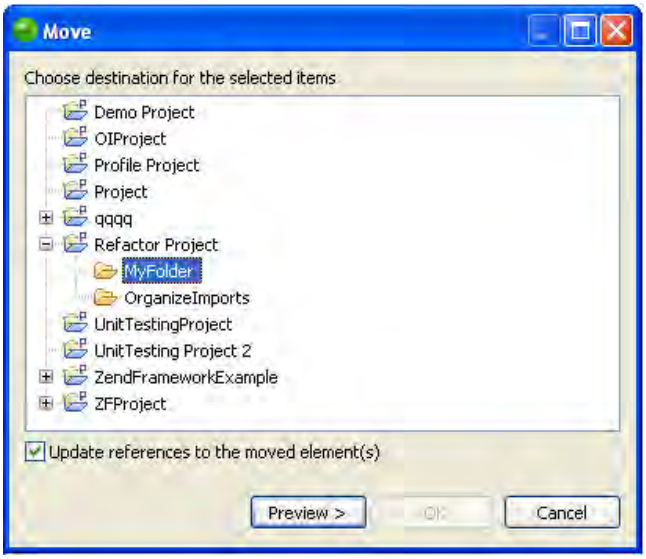
This procedure describes how to move a file, which will result in the automatic updating of all instances where that file is referenced (required, included etc.) within the project to reflect its change of location.

Note:

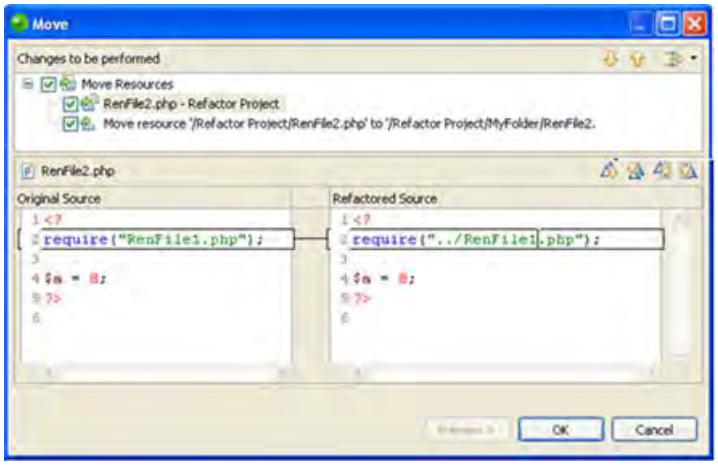
Ensure that you save any changes to the file before applying the refactoring feature.

To move a file:

1. In PHP Explorer view, right-click the file which you would like to rename and select Refactor | Move -or- press Alt-Shift-V.
   A Move File dialog will appear.



2. Select the new location of the file.
3. Check the "Update references" box and click Preview.
   A preview window with a changes tree will open showing all the changes which will be made to reflect the move of the file.



4. You can scroll through the different changes using the scrolling arrows .
   Note that, if the file has been referenced (required, included etc.) in other files, the reference to the location of that file in other files will also have been changed.
5. Unmark the checkbox of changes which you do not want applied.
6. If you are satisfied with the changes, press OK.

The file will be moved and the file's new location will be updated in all instances where that file is referenced.

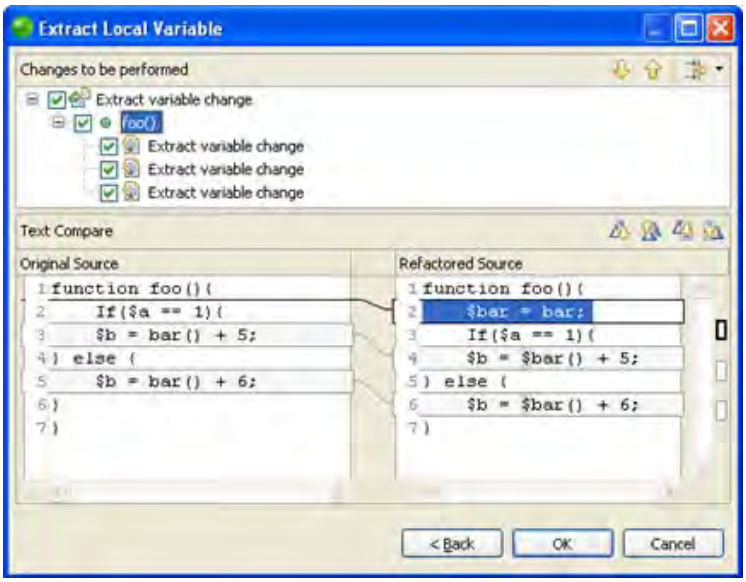
Extracting Variables

The extract variable feature can create a local variable to replace all occurrences of a given expression.

To create a new variable from an expression:

1. Place your cursor on the expression which you would like to replace with a variable.
2. Right-click and select Refactor | Extract Variable
   -Or- press Alt+Shift+L.
   The Extract Local Variable dialog is launched.

3. Enter the name of the new variable in the Variable name field.
4. Mark the checkbox so that all occurrences of the selected expression will be replaced by references to the local variable.
5. Click OK to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
6. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the extracting of the variable.



7. The changes will be listed according to the context within which they appear. You can expand the nodes to see all changes within particular files, classes or functions.
8. Use the Next / Previous Change arrows to scroll through all possible changes. Unmarking the checkboxes next to the changes will cause those changes not to take effect.
9. Click OK to apply the changes.

The variable will be extracted and the relevant changes made to the code.

Extracting Methods

The extract method feature can create a method to replace all occurrences of a given code fragment.

To create a new method from an expression:

1. In the editor, select the the code fragement which you would like to replace with a method.
2. Right-click and select Refactor | Extract Method -or- click Alt+Shift+M.
   The Extract Method dialog is launched.

3. Enter the name of the new method in the Method name field.
4. Select the Access modifier for your method.
5. If multiple occurrences of the code fragement appear in your code, mark the 'replace all occurrences of statements with method' checkbox for all occurrences to be replaced with the new method.
6. Mark the Generate method comments checkbox for comments to be created for your method.
7. Click OK to apply your changes or click Preview if you want to see a preview of the changes that this refactoring will create.
8. If you clicked preview a preview window will open with a changes tree showing all the changes which will be made to reflect the extracting of the method.
7. The changes will be listed according to the context within which they appear. You can expand the nodes to see all changes within particular files, classes or functions.
8. Use the Next / Previous Change arrows to scroll through all possible changes. Unmarking the checkboxes next to the changes will cause those changes not to take effect.
9. Click OK to apply the changes.

The method will be extracted and the relevant changes made to the code.

Generating Getters and Setters

Zend Studio can automatically create getter and setter functions in order for 'Get' and 'Set' function calls to be easily created.

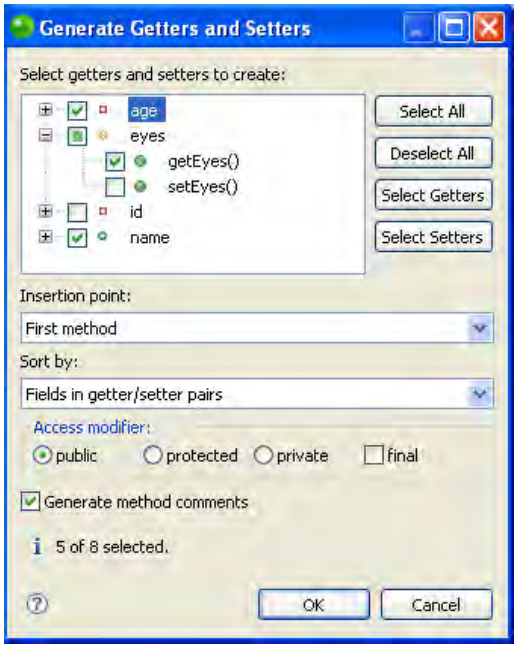
This procedure describes how to generate getter and setter functions for all variables within a class.

To generate getters and setters:

1. In a PHP file, place your cursor within the class for which you would like to generate the getters and setters.
2. Right-click within the class's source code and select Source | Generate Getters and Setters -Or- from the Menu Bar go to Source | Generate Getters and Setters
   -Or- Right-click the required class in PHP Explorer view and select Source | Generate Getters and Setters

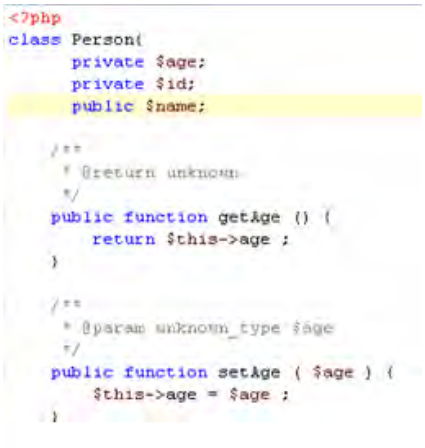
```
1  <?php
2  class Person{
3          private $age;
4          private $id;
5          public $name;
6          protected $eyes;
7  }
8
9  ?>
10
```

3. The 'Generate Getters and Setters' dialog will open, displaying all variables that might require a getter/setter.



4. Select:
   - Which variables getter and setter functions should be created for.
     Expand the list under each function by clicking the + icon in order to select to generate only a getter or a setter for each variable.
     If your cursor was originally placed on a certain variable, this will automatically be selected.
   - Insertion Point - Select the location in the class where you want the entries to be added from the drop-down list.
     Options are:
     - First method - Getters and setters will be placed as the first methods within the class.
     - Last method - Getters and setters will be placed as the last methods within the class.
     - Cursor position - Getters and setters will be placed at the cursor position (only available if cursor was placed in a valid position).
     - After function ... - Getters and setters will be placed after the selected function (depending on the functions available within the class).
     - Sort by - Determine the order in which the entries are entered.
       The options are:
     - First getters, then setters - All the getters will be grouped together, followed by the setters.
     - Fields in getter/setter pairs - Pairs of getters and setters relating to the same variable will be generated together.
   - Access modifier (Not available in PHP 4 projects) - Selects whether the functions will be public, protected, private or final.
   - Generate method comments - Select whether to generate a PHP Docblock for each entry.
5. Click OK.

The relevant getter and setter functions will be generated for the selected functions.



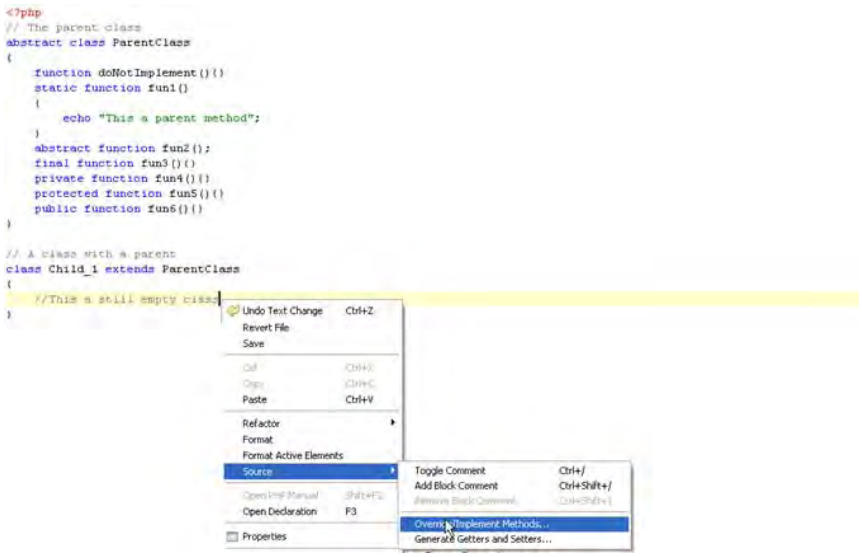
```php
<?php
class Person{
      private $age;
      private $id;
      public $name;

    /**
     * @return unknown
     */
    public function getAge () {
        return $this->age ;
    }

    /**
     * @param unknown_type $age
     */
    public function setAge ( $age ) {
        $this->age = $age ;
    }
```

Overriding / Implementing Methods

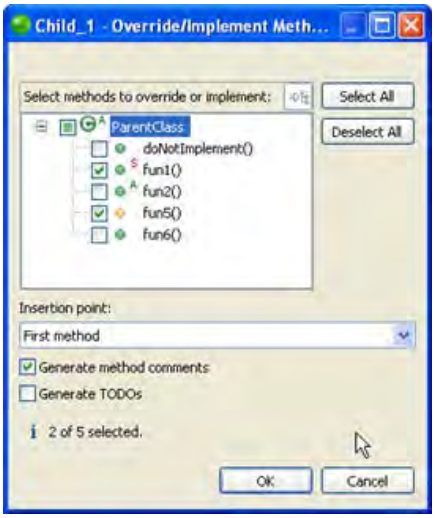
The override / implement method feature provides a mechanism for overriding/implementing methods defined in a class's parent or interface.

To override / implement methods:

1. Place your cursor within the class for which you would like to override/implement methods.

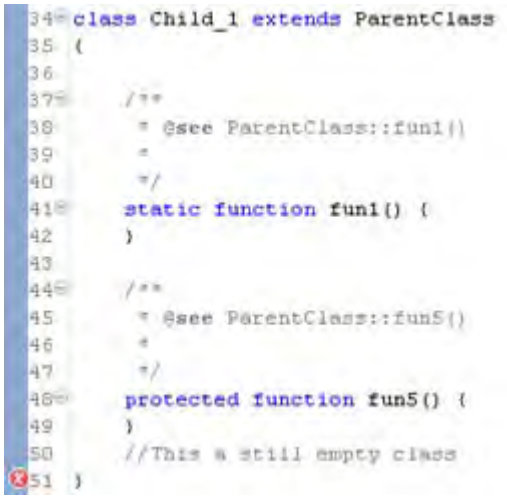


2. Right-click within the class's source code and select Source | Override/Implement Methods -Or- Right-click the required class in PHP Explorer view and select Source | Override/Implement Methods.
   The ' Override/Implement Methods' dialog will open, displaying all methods that can be overridden.



3. Select the elements which you would like to override or implement by marking the checkbox next to the element.
4. Select the insertion point from the drop-down list.
5. Select whether to generate method comments or TODOs (tasks) by marking the relevant checkboxes.
6. Click OK.

The relevant code to implement / override the selected methods will be created in your script.

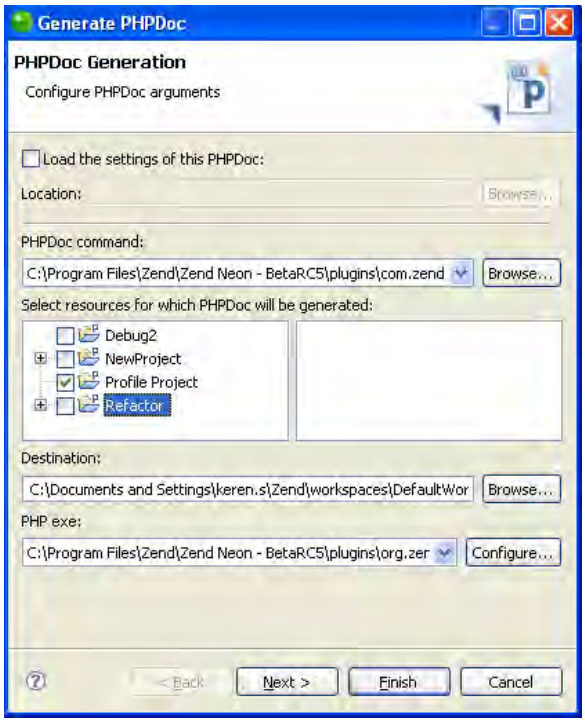


Creating a PHPDoc

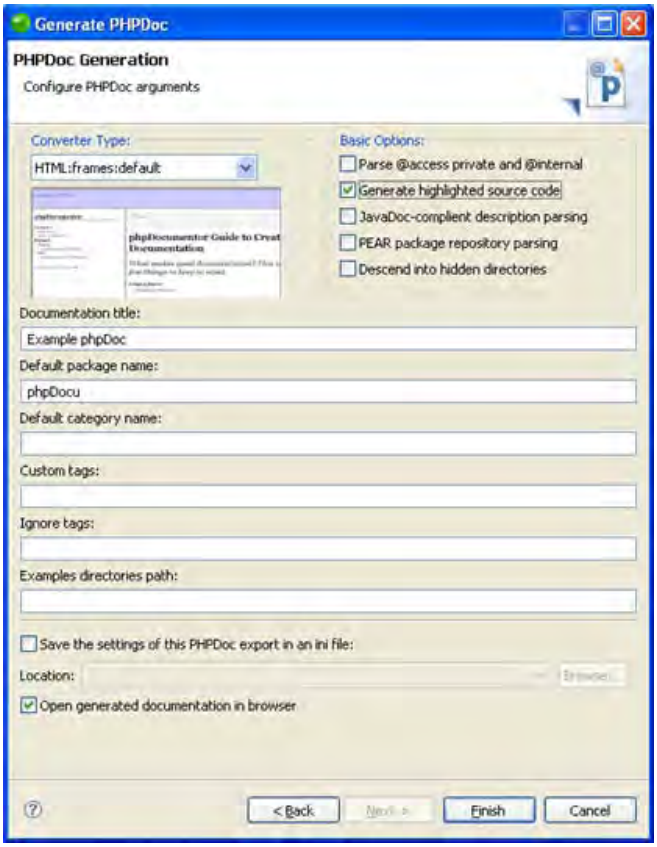
This procedure describes how to create a PHPDoc from your PHP files.

To create a PHPDoc:

1. Right-click the project from which you would like the PHPDoc to be generated and select Generate PHP Doc -or- go to Project | Generate PHPDoc -or- press Alt+D. The PHPDoc Generation dialog will open.



2. If you have previously created PHPDoc settings which you would like to apply, mark the checkbox. To create a settings configuration, see point 8.
3. Ensure that the required project, destination for the PHPDoc and PHP Executable are selected.
4. Click Next.



5. Choose the style for your PHPDoc from the 'Converter Type' drop-down list. This will affect the layout and format of your PHPDoc.
6. Select which basic options you want to apply: (Refer to the phpDoc Manual online at http://www.phpdoc.org for complete descriptions of the options.)
   - Parse @acess private and @internal
   - Generate highlighted source code
   - JavaDoc-compliant description parsing
   - PEAR package repository parsing
   - Descend into hidden directories
7. Enter the following fields: (Refer to the phpDoc Manual online at http://www.phpdoc.org for complete descriptions of the options.)
   - Default package name
   - Default category name
   - Custom tags (if required)
   - Ignore tags (if required)
   - Examples directories path (if required)
8. To save these settings so that they can be reused when creating new PHPDocs,

mark the 'Save the settings of this PHPDoc export in an ini file" checkbox and specify where the ini file should be saved.

9. Ensure the 'Open generated documentation in browser checkbox is marked to view your PHPDoc once created.
10. Click Finish.

    A Running PHPDocumentor dialog will appear.

Your PHPDoc will be automatically created and will be opened in a browser.

By default, your phpdoc is created as an index.html file in a folder entitled 'docs' in the root of you Workspace. (e.g. C:\Documents and Settings\bob\Zend\workspaces\DefaultWorkspace\docs\index.html).

Creating HTML files

These procedures describe how to create new HTML files, allowing implementation of Zend Studio's full HTML editing functionality.

To create a new HTML file:

1. In PHP Explorer view, select the folder into which you would like to create the file and from the Menu Bar go to **File | New | HTML Page** -or- right-click the folder and select **File | New | HTML. Page**.
   The new HTML Page dialog will appear.
2. Enter the file's name and click **Next**.
3. Select which HTML template to use for the new file and click **Finish**.

The new HTML file will be created and will open in a standard HTML editor.

If you are interested in editing your code in a WYSIWYG view, you can install the WST Web Page Editor plugin. This plugin provides functionality while allowing you to view something very similar to your end outcome before the project has been completed. To download the necessary plugin, follow this link: http://kb.zend.com/index.php?View=entry&EntryID=419.
For more information on adding a plugin see Update Manager.

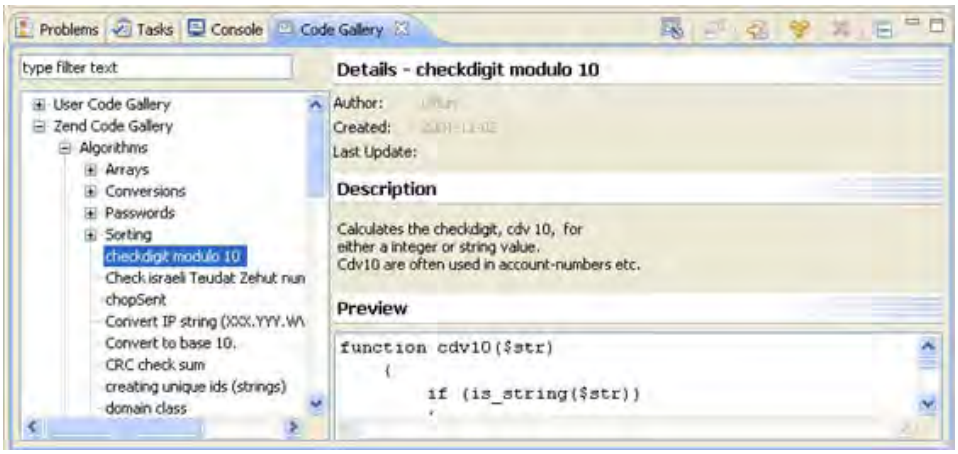
Note:

More information about writing HTML can be found at: http://www.w3schools.com/html

Using Code Galleries

The Code Gallery view gives you access to code snippets that you pre-defined or that are available through a Code Gallery site.

The Code Gallery view can be accessed from Window | Show View | Code Gallery.



See Inserting Code Snippets into your Script, Creating and Editing Code Gallery Entries, and Interacting with Code Gallery Sites for more on working with Code Galleries.

Inserting Code Snippets into your Script

This procedure describes how to insert existing code snippets into your script.

To insert a code snippet into your script:

1. Open the Code Gallery view by going to Window | Show View | Code Gallery.
2. Place the cursor at the place in the editor into which you would like the code snippet to be entered.
3. Expand the nodes next to the required User Code Gallery in the Code Gallery view -or- enter a string into the 'type filter text box' to search for a particular category or snippet.
4. Select the required code snippet.
   The details of the code snippet, including its description and a preview of the code, will appear in the right-hand pane.
5. Right-click the snippet and select Insert -or- click the Insert button on the view's toolbar.

The selected snippet will be entered into your script.

Note:

Many snippets contain PHP tags. If necessary, ensure that you remove existing PHP tags from your script to avoid duplication.

Creating and Editing Code Gallery Entries

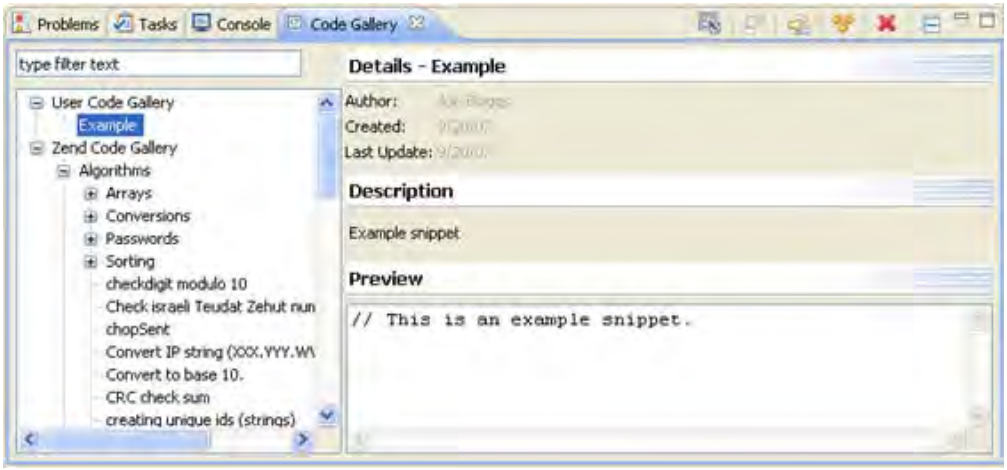
This procedure describes how to create a new code snippet and add it to your User Code Gallery so that you can quickly and easily insert it into your script.

To add a code snippet to your list:

1. Open the Code Gallery View by going to Window | Show View | Code Gallery.
2. Right-click the 'User Code Gallery' from the list and select New Entry -or- click the New Entry icon on the Code Gallery view's toolbar.
   A New code gallery entry dialog will open.

3. Enter the code snippet's name, author and description in the relevant fields.
4. In the 'Code' box, enter the code snippet.
5. Click OK.

Your new code snippet will be added to the list.



To edit an existing snippet:

1. Open the Code Gallery View by going to Window | Show View | Code Gallery.
2. Expand the nodes next to the required User Code Gallery in the Code Gallery view - or- enter a string into the 'type filter text box' to search for the particular category or snippet.
3. Select the required code snippet.
   The details of the code snippet, including its description and a preview of the code, will appear in the right-hand pane.
4. Right-click the snippet and select Edit.
5. The New code gallery entry dialog will open containing the snippet's details.

6. Make the required edits and click OK.

The code snippet will be updated with the edits you have made.

Note:

You can only edit snippets in the User Code Gallery.

Interacting with Code Gallery Sites

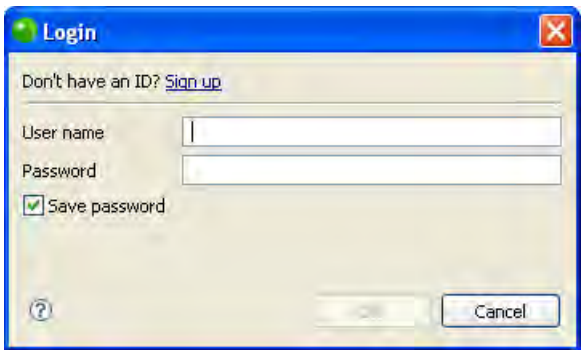
The Code Gallery view allows access to Code Gallery sites that contain a variety of code snippets.

These procedures describe how to connect to the Zend Code Gallery site to download Zend's code snippets,how to add a new Code Gallery site, how to update your Code Gallery list, how to suggest code snippets you have created to the site for use by others and how to give a rating to downloaded code snippets.

To access the Zend Code Gallery:

1. Open the Code Gallery view by going to Window | Show View | Code Gallery.
2. Expand the node next to the Zend Code Gallery list.

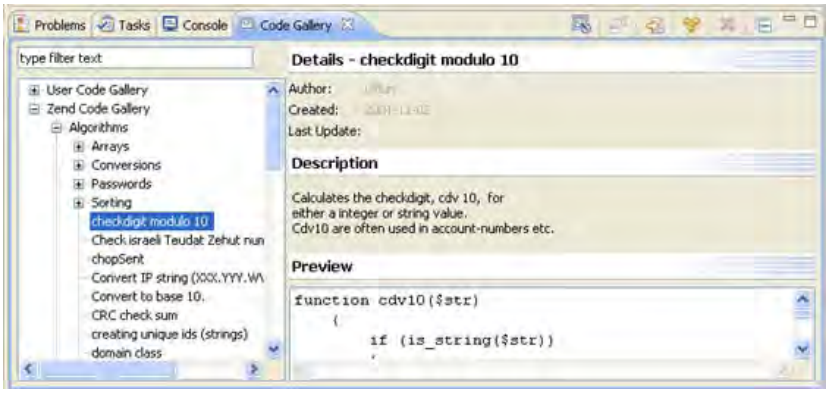
   If no node appears, click the 'synchronize with site' button to reactivate it.
3. A login dialog will appear.



4. Enter your Zend Network User name and Password.

   If you don't have a Zend Network ID, click the 'Sign up' link to be taken to the Zend Developer Zone registration site, or follow this link: https://devzone.zend.com/member/register.

Your Zend Code Gallery list will be updated with all the code snippets from the Zend Code Gallery site, divided into categories.



To add a Code Gallery site to the Code Gallery list:

1. Open the Code Gallery preferences page by going to Window | Preferences | PHP | Code Gallery, or clicking the 'Configure Code Gallery' button on the Code Gallery view's toolbar.
2. Follow the instructions under 'Adding a Code Gallery' in the 'Code Gallery preferences' help page.

To update your Code Gallery:

1. Open the Code Gallery view by going to Window | Show View | Code Gallery.
2. Select the Code Gallery which you would like to update.
3. Click the 'synchronize with site' button

Your code gallery list will be updated with all the latest changes from the Code Gallery site.

To suggest that a code snippet you have created be added to a Code Gallery site:

1. Open the Code Gallery view by going to Window | Show View | Code Gallery.
2. Expand the User Code Gallery node and right-click the code snippet you have created.
3. Click suggest.

   The Suggest to Code Gallery dialog will appear.

4. Select the Code Gallery to which you would like to suggest your snippet from the Code Gallery drop-down list.
5. Select the category to which you would like to associate it.
6. Click OK.

Your code snippet will be sent to the chosen site for consideration.

To rate a snippet which you have downloaded from a Code Gallery:

1. Open the Code Gallery view by going to Window | Show View | Code Gallery.
2. Expand the relevant Code Gallery node and right-click the required code snippet.
3. Select 'Rate...'.

   The 'Send your rating' dialog will appear.

4. Select your rating for the snippet (1 being the lowest and 5 being the highest).
5. Press OK.

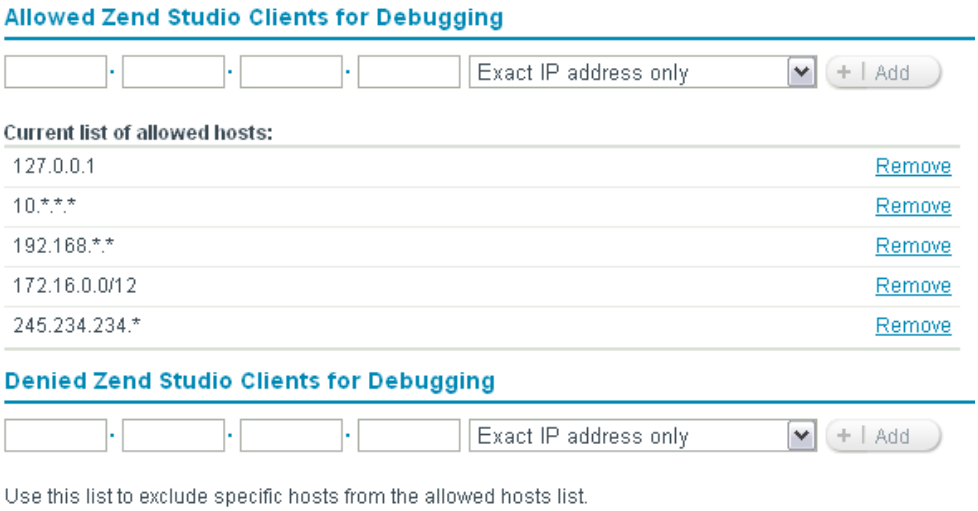
Your rating will be sent to the relevant Code Gallery site.

Configuring Studio Communication Settings in Zend Server

In order for you to be able to view, debug and profile Zend Server Events in Zend Studio, you must ensure the correct communication settings are configured in your Zend Server.

To configure Zend Studio communication in Zend Server:

1. Open your Zend Server GUI.
2. Go to the Server Setup | Debugger tab.



3. Ensure the address of your Zend Studio is included in the Allowed Hosts sections. This will ensure you can debug/profile Zend Server Events.
   To add an address to the list:
   i. Enter the IP address or Net mask of the machine on which your Zend Studio is installed.In Order to enter a Net mask, enter a range by entering the beginning of an IP address and adding '0' instead of the rest of the number. To make sure you are using Wildcards (*) to specify a range of IPs select the pattern you want from the drop-down list.
   ii. Click Add. Your Zend Studio machine's address will be added to the Allowed Hosts list.
4. Ensure your Zend Studio's IP address is not in the Denied Hosts list.
   If it is, click Remove next to the required address to remove it from the list.
5. In the Zend ServerGUI,go to the Server Setup tab | Monitor and configure the following:
   - Auto detect theZend Studio Client Settings - Set to 'On' to inform Zend Server of the method of connection to Zend Studio. This allows Zend Server to

automatically detect your Zend Studio Debug settings.

6. Click Save.
7. Restart your Web server for the settings to take effect.

Defining Zend Server in Studio

Integration of Zend Server with Zend Studio allows the appliance of Zend Studio functionality (Profiling, Debugging etc.) to Server Events, as well as allowing access to Zend Server's Event list.

Before configuring Zend Server in Zend Studio, Zend Server must be installed and running.

To define Zend Server:

1. Open the PHP Servers Preferences page by going to Window | Preferences | PHP | Editor | PHP Servers.
2. Click New to create a new server with Zend Server Integration or select an existing server and click Edit to add Zend Server integration to a previously configured server.
   A PHP Server Creation dialog will open.
3. Configure the server as described in the PHP Server Preferences page (enter the Server's name and document root's URL.)
4. Click Next.
5. If necessary, define Path Mapping. See Adding a Server Location Path Map for more information.
6. Click Next.
   A Zend Server Integration dialog will appear.
7. Mark the Enable Zend Server Integration checkbox to enable Zend Server Integration features.
8. tInsert the desired Zend Server GUI URL suffix and Port Number.
   Leaving the Use default checkbox marked will create a URL in the format <server's document root>/ZendServer>. If necessary, unmark the checkbox to edit the suffix to point to your Zend Server GUI URL.
9. Enter your Password.
10. Click Next to configure Tunneling settings or Finish to create your server.

Zend Server will be added to the Server list and will allow you to use Zend Server integration features.Zend Server will also now be available from the toolbar and the Zend Server Events view.

Integrating with Zend Guard

Zend Guard's Integration with Zend Studio allows you to:

- Encode your Zend Studio projects using Zend Guard.
- Open and edit Zend Guard projects in Zend Studio

See the Zend Guard product site (http://www.zend.com/en/products/guard) or the Zend Guard Online Documentation (http://files.zend.com/help/Zend-Guard/zend-guard.htm) for more information on Zend Guard.

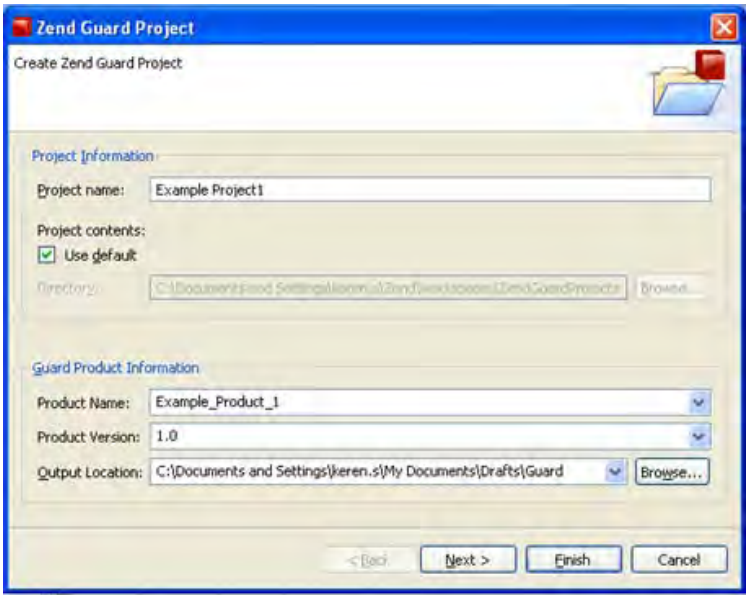
Encoding Projects Using Zend Guard

This procedure describes how to open your Zend Studio Projects in Zend Guard in order to encode them.

Zend Guard must be configured in Zend Studio before Zend Guard integration is accessible. This can be configured through the Zend Guard preferences page, accessible from Window | Preferences | PHP | Zend Guard.

To open a project in Zend Guard:

1. In PHP Explorer view, right-click the required project and select Encode Project -or- select the required project and from the Menu Bar go to Project | Encode Project -or- click the Encode Project button on the toolbar.
   If you have not configured your Zend Guard location in Zend Studio, a prompt will appear stating that Zend Guard preferences have not been defined. Click on the 'Define Zend Guard.exe' link to be taken to the Zend Guard preferences page or go to Window | Preferences | PHP | Zend Guard.
   If you have defined your Zend Guard location, Zend Guard will now be opened and a Zend Guard Project dialog will open.



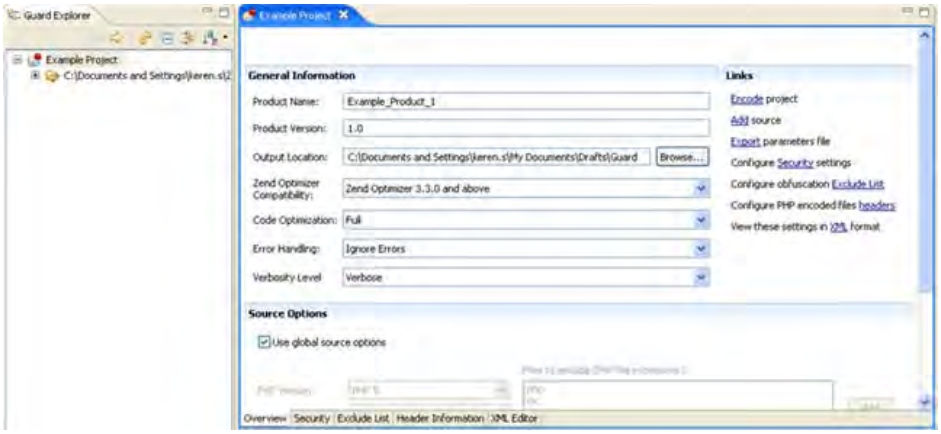
2. Enter the following information in the relevant fields:
   - Project Name - This will automatically be taken from the name of your Zend Studio project.

This cannot be the same name as an existing project in Guard.

- Project contents - The directory into which the project will be placed. Unmark to browse to a different location.
- Product Name - Enter the product's name.
- Product Version - Enter the product's version.
- Output Location - Enter the location to which you would like the encoded files to be created.

3. Click Next.
4. If you want additional files and folders to be added to the project, click Add Folder or Add File and browse to the required source.
5. Click Next.
6. Select:
   - The PHP version
   - Whether to enable Short Tag Support - Enable recognition of short PHP tags. Recognizes <? as a valid PHP start tag. When this option is not selected, Zend Guard will not encode short tags, which will be treated as regular HTML.
   - Whether to enable ASP Tag Support - Enables recognition of ASP tags. Recognizes <% as a valid PHP start tag. When not selected, code within ASP tags is treated as regular HTML.
   - Resolve Symlinks - Resolves Symbolic Links before encoding (not applicable in Windows). A symbolic link (often shortened to symlink and also known as a soft link) consists of a special type of file that serves as a reference to another file or directory. Unix-like operating systems in particular often feature symbolic links.
   - Which files to encode - Lists the file extensions for Guard to encode (extensions not listed will not be encoded). File extensions that are not listed here and in "Patterns to Ignore" will be sent as-is to the output folder.
   - Which patterns to ignore - Files matching these patterns will not be encoded when encoding a directory, nor will they be copied as-is to the target directory. By default, the list contains the CVS directory and cvsignore files (includes Wildcards '*').
7. Click Finish.

Your project will be opened in Zend Guard and can be encoded using Zend Guard's functionality.



See the Zend Guard Online Help for more information on encoding your projects.

Note:

Your production server must be running the Zend Optimizer in order for your PHP to be able to run the files encoded by Zend Guard. The Zend Optimizer comes bundled with Zend Server or Zend Core or can be downloaded from the Zend Guard product page (http://www.zend.com/en/products/guard/optimizer).

Note:

Always test encoded files before uploading them to your production server.

Opening and Editing Zend Guard Projects in Zend Studio

This procedure describes how to open Zend Guard's file in Zend Studio, enabling users to extend Zend Guard with the rich editing features included in Zend Studio. This provides users with a seamless workflow for fixing and modifying code through Zend Guard.

To open a Zend Guard file in Zend Studio:

1. In Zend Guard, go to Edit | Preferences | Zend IDE.
2. Enter the path to Zend Studio and click Apply and OK.
3. Right-click the required file in Guard Explorer view and select "Open with Zend IDE." This option will only be available once Zend Studio integration has been configured as described in steps 1 and 2.
4. Zend Studio will open and the file will be displayed in an editor.
   All of Zend Studio's editing capabilities will now be available to the file.
5. Once you have made the required edits, click Save on Zend Studio's Menu Bar to save the edits made to the file. Changes will be updated in the file accessible through Zend Guard.

Working with WSDL

Using Zend Studio, you can reference and view existing WSDL files in your PHP Code.

See Incorporating WSDL Files for more on referencing WSDL files in your code.

Incorporating WSDL Files

Incorporating a WSDL file is the process of taking a service or services from an existing WSDL file and integrating their capabilities (functions) into your PHP code.

To reference a WSDL file and benefit from full integration into Zend Studio (code completion and function display in the PHP Project Outline tab) the file must be present in your local file system (Method 1), or can be referenced by using the WSDL file's URL (Method 2).

To create a SOAP Client:

Method 1

1. Use an external browser to open the URL.
2. Save the contents of the URL as a WSDL file. e.g "C:\GoogleSearch.wsdl".
3. Create the soap client by referencing the WDSL file in your PHP script.

   e.g:

```
$a = new SoapClient("C:\GoogleSearch.wsdl")
```

The WSDL file's methods, classes, functions, etc. will now be available in the Code Assist Menu and the SOAP Client now appears in the PHP Project Outline tab.

To create a SOAP Client:

Method 2

1. Create a new SoapClient instance in your PHP code, containing the URL of the WSDL file which you want to incorporate.

   e.g.:

```
$a = new SoapClient("http://api.google.com/GoogleSearch.wsdl");
```

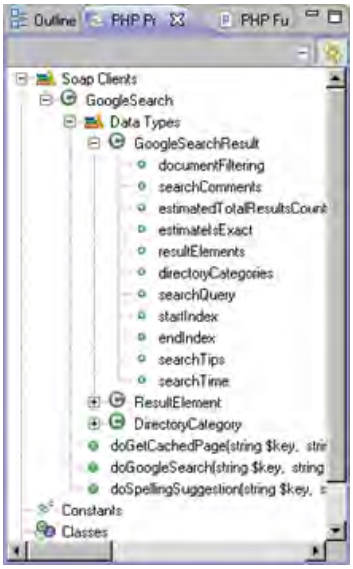
2. Save the file.

The WSDL file's methods, classes, functions, etc. will now be available in the Code Assist Menu and the SOAP Client now appears in the PHP Project Outline tab.

Once a WSDL file has been referenced in your project, the following will be affected:

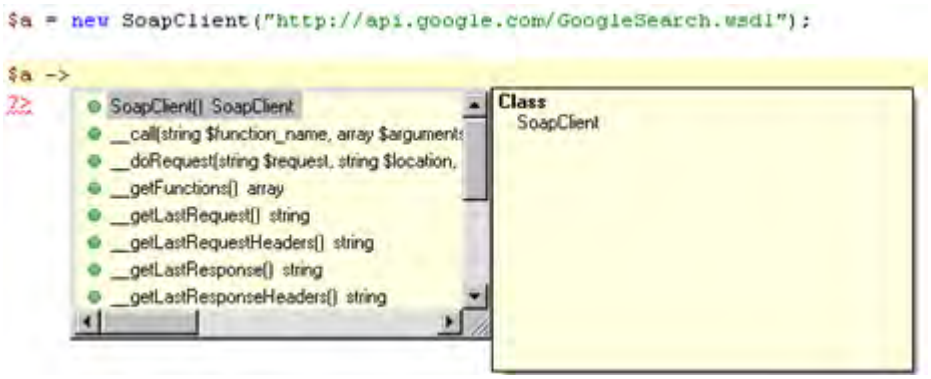
Outline and PHP Project Outline views

The PHP Project Outline will now include all the functions from the referenced WSDL file.



Content Assist

Code completion is automatically updated with all the functions included in the referenced WSDL file.



Note:

Code completion for Web Services is supported for PHP 5.

Auto Link to WSDL files

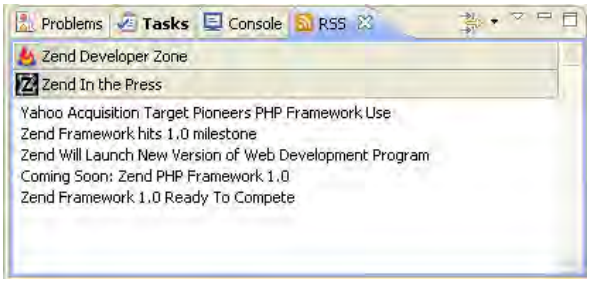
Transform the name of the referenced WSDL file into a link by hovering over the file's name and pressing CTRL. Clicking the link (while CTRL is still pressed) will jump to the WSDL file if it is already open in the editor. Auto Link to Files exists for every string containing a file name in the editor.

Viewing RSS Feeds

These procedure describe how to view RSS feeds within Zend Studio's RSS view and how to add additional RSS channels.

To view RSS feeds:

1. Open the RSS view by going to Window | Show View | RSS.

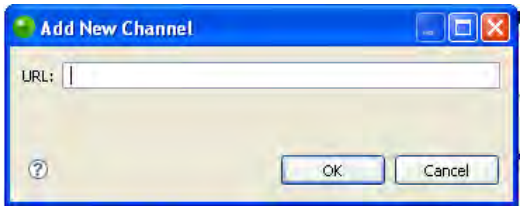

Problems
Tasks
Console
RSS
Zend Developer Zone
Zend In the Press
Yahoo Acquisition Target Pioneers PHP Framework Use
Zend Framework hits 1.0 milestone
Zend Will Launch New Version of Web Development Program
Coming Soon: Zend PHP Framework 1.0
Zend Framework 1.0 Ready To Compete

2. The RSS reader comes pre-loaded with the Zend Developer Zone feed and the Zend In the Press feed.
3. Click the required heading to expand the list and see all the news items underneath it.

   You can group the items by channels or by time by clicking the view's menu icon and selecting the relevant option (Group by Channels / Group by Time).
4. Double-click the item you want to view to open it in Zend Studio's internal browser.

To add an RSS channel:

1. Open the RSS view by going to Window | Show View | RSS.
2. Click the view's menu icon and select Subscribe.
3. The Add New Channel dialog will be displayed.



4. Enter the URL of the site from which the RSS feeds will come and click OK.

The new RSS feed will be displayed in your RSS view.

Working with Remote Server

Working with Remote Server

Remote Server allows you to set up a remote server connection, which will help you to create, view and manage your connections to remote systems such as FTP and SSH.

The following tasks will guide you through the actions that can be done by using the Remote Server functionality:

- Creating a Connection via the Server View
- Creating a New Remote Project
- Downloading a Project from an Existing Connection
- Sharing a Local Project to a Connection
- Synchronizing a Local Project to a Connection
- Importing Selected Files from a Folder

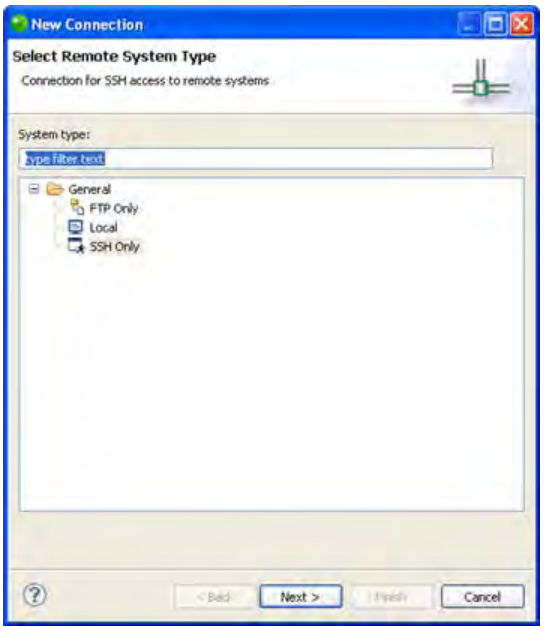
Creating a Connection via the Server View

The Server View allows you to setup different types of connections, for creating and managing projects.

This procedure describes how to create a connection via the Server view, in order to view the current status of connection and establish connection to a remote server.

To create a connection via the Servers View:

1. Open the Servers View, situated at the bottom of the page.
2. Right-click the Connections node and click **New | New Connections** to open the New Connections dialog box.



3. Use the New Connection dialog to choose the connection type by clicking on the location or use the '**Type Filter Text**' box.The connection options are:

- FTP Only – Connect to the FTP access to remote systems.
- Local – Connect to your local file system on your computer.
- SSH Only – Connect to an SSH to access remote server.

4. Click **Next** to define the system connection, and enter the following details:

- Parent Profile – Your computer profile.
- Host Name – Host Name or IP address of the target system.
- Connection Name – A unique name that is attached to the profile.
- Description – Connection description that will help you to remember what this connection is.

5. Tick the ‘Verify Host Name’ box if you want to validate your connection now. Zend Server will attempt to connect to the remote server. If the connection fails this can mean that the server is not available or the information entered is incorrect, retry or Click **Next** to continue.
6. For FTP or Local connection types Click **Finish** to add your new connection.
7. For SSH connection types, you will now have the option to set one of the following system checks:

- SFTP Files – Work with files on remote systems using the secure Shell (SSH) protocol.
- SSH Shells - Work with shells and commands on remote systems using the Secure
- Shell (SSH) protocol.
- SSH Terminals - Work with terminals and commands on remote systems using the Secure Shell (SSH) protocol.

To set system checks on any of these options do the following:

1. Go to the Available Services box in the dialog and select one of the following options.

- SSH/SFTP file services.
- SSH connector service.

2. Click on the SSH connector service, to expand the node.
3. Click on SSH Settings.

Use the properties table, to edit your options:

- Keepalive (secs) – At regular intervals, the system checks to see that the connection is alive. If so, you will then receive a server notification.
- Timeout (secs) - At regular intervals, the system checks to see if the connection has timed out, and if so you will be sent a server notification.

4. Change the scheduled checks in the SSH Settings, in the properties table by clicking on the value of the relevant option and inputting how many seconds apart you want your checks to run.
5. Click **Finish** to add your new connection and to set your system checks.

Now that you have completed creating your connection via the server view, you will be able to work remotely on new and existing projects to import, export and synchronize projects.

Creating a New Remote Project

This procedure describes how to create a new remote project. Once the remote server has been set up, you can create a new project which will be synchronized with one of the remote servers. Before creating a new remote project make sure that you have already created a Creating a Connection via the Server View .

To create a new remote project:

1. From the menu bar, go to **File | New | PHP Project** -Or- In PHP Explorer view
2. Right-click and select **New | PHP Project.**
3. The New PHP Project wizard is displayed.
4. Go to the **Remote Server** section.

4. Tick the box entitled **Enable the Remote Web Site**:
5. Browse for your location.

6. Enter the following information:
   - Project name - The required project name.
   - Contents – There are three options. Choose the one relevant to your needs.
     - Create a new project in the workspace - Creates a new PHP project in the workspace directory.
     - Create a project from an existing source - Creates a PHP project pointing to files situated outside of the workspace. Click Browse to select the required source content.
     - Create project on a local server - Creates the project on a local server. This option will only be available if a local Zend Server has been configured in the PHP Servers Preferences page.
     - PHP Version - Select whether to use the default PHP version settings or unmark the 'Enable project specific settings' checkbox to configure the PHP version used for this project.

   See PHP Interpreter Preferences for more information.
   - Use default PHP settings - Uses the default PHP Interpreter settings.
   - Use project specific settings - Select the PHP version to be used for the project and whether ASP tags are used as PHP tags.

   See PHP Version Support for more information.
   - Project Layout - Select whether to:
     - Use project as source folder - All resources within the project will be added to the Build Path by default.
     - Create separate folders for source files and public resources - Separate folders will be created in which you can place resources which should be included or excluded from the Build Path. See Configuring a Project's PHP Build Path for more information.
       The default setting for this option can be configured from the New Project Layout Preferences page.
     - JavaScript Support - Mark the 'Enable JavaScript support for this project checkbox for JavaScript functionality (e.g. JavaScript Content Assist options) to be available to the project. See Enabling JavaScript Support in PHP Projects for more information.
7. Click **Next** to configure the project's Include Path (this can also be done following the project creation).

8. Click **Next** to configure the project's PHP Build Path (this can also be done following project creation).

9. Click **Finish**.

The new PHP project will be created in your workspace and displayed in PHP Explorer View. You can now start to develop your application by creating PHP Files or adding other resources to your project.

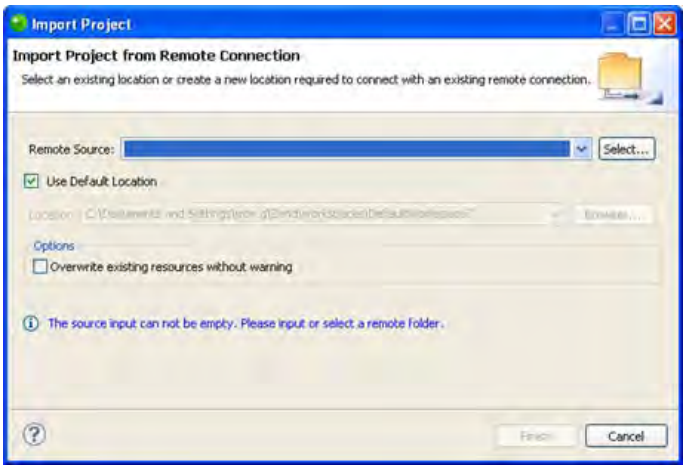
Downloading a Project from an Existing Connection

This procedure describes how to download a project from an existing connection, therefore making a local files available. Before downloading a project from an existing connection, you must have already created a remote server connection (Creating a Connection via the Server View).

To download a project from an existing connection:

1. Go to the **Servers View**.
2. Double-click **Connections**.
3. Right-click the relevant connection and click **Import Project**.
4. Select the **Remote Source** location.
5. Click **Finish**.

   The project will appear on your workspace.


Import Project
Import Project from Remote Connection
Select an existing location or create a new location required to connect with an existing remote connection.
Remote Source:
Select...
Use Default Location
Options
Overwrite existing resources without warning
The source input can not be empty. Please input or select a remote folder.
Cancel

Alternatively, you can gain access via the main menu.

1. Click the file tab on the main menu.
2. Click **Import**.
3. Double-click the **Remote Connections**.
4. Double-click **Remote Project** and the 'Import Project' dialog will open up.
5. Click **Select** to locate your Project from the Remote Connections folder.
6. Double-click on your project.
7. Click on **Finish.**

   The project will be created and appear on your workspace.

Sharing a Local Project to a Connection

This procedure describes how to share a local project to a connection. Before sharing a local project to a connection, you must have already created a remote server connection (Creating a Connection via the Server View).

To share a local project to a connection:

1. Right-click on the un-shared project in the workspace.
2. Select **Team**.
3. Click on **Share Project**.
4. The **Share Project** wizard will open, select the **Remote Server** node and click Next.
5. Ensure that the correct project has been selected. If not, **Browse** for the folder you will be uploading the project to, and select it.
6. Click **Finish**.

Your shared project will now appear on your workspace, identified with an icon.

Synchronizing a Local Project to a Connection

This procedure describes how to synchronize a local project to a connection. Before synchronizing a local project to a connection, you must have already created a remote server connection.

To synchronize a local project to a connection:

1. Right-click on the shared project.
2. Select **Team**.
3. Select **Synchronize**.
4. Double-click on the shared file, in order to automatically open the **Compare Editor**.

You will now see your project on your workspace.
If there are any conflicting files, double click on the file and the Compare Editor will appear. The synchronized view will reveal any differences there are between the local and shared files, by using these icons:

	An incoming addition means that a resource has been added to the repository. Updating will transfer the resource to your workspace.
	An incoming change means that the file has changed in the repository. Updating will transfer the new file revision to your workspace.
	An incoming deletion means that a resource was deleted from the server. Updating will delete your local resource.
	An outgoing addition means that the file was added to your workspace and is not yet in the repository. Adding then Committing will transfer the new file to the repository.
	An outgoing change means that the file was change locally. Committing the file will transfer the changes to the repository and create a new revision of the file.
	An outgoing deletion is a resource that has been deleted locally. Committing these resources will cause the remote resource to be deleted. Note: in CVS directories are never really deleted from the repository. Instead, files are deleted and empty directories are pruned from your workspace.
	A conflicting additions means that the resource has been added locally and remotely.
	A conflicting change means that the file has been changed locally and remotely. A manual or automatic merge will be required. Also, any entries in the view that contain children that are conflicts will also be decorated with the conflict icon. This is done to make conflicts easy to find.
	A conflicting deletion means that the resource was deleted locally and remotely.

Upload to a Server

This procedure describes how to upload resources to a server. This action enables you to move files to a local connection. Before uploading, you must have already created a remote server connection and synchronized the project that contains the resource/s.

To export resource/s via PHP Explorer:

1. Select the chosen resource/s in PHP Explorer.
2. Right-click on the resource and from the right-click menu select | **Team | Export**.

The chosen resource/s will be moved to the remote location.

You can also export resources via the Synchronize View.

To export files via the Synchronize View:

1. Select the chosen resource/s in the Synchronize View.
2. Right-click on the resource and from the right-click menu select **| Team | Export.**

The chosen resource/s will be moved to the remote location.

Download from a Server

This procedure describes how to download resources from a server. Before downloading, you must have already created a remote server connection and synchronized the project that contains the resource/s.

To import resources:

1. Select the resource/s in the **Synchronize View**.
2. Right-click on the resource and from the right-click menu, select **Import**.

The resource/s will appear in your workspace.

You can also import resources via PHP Explorer:

1. Select the resource/s in PHP Explorer.
2. Right-click on the resource and from the right-click menu, select **Import**.

The resource/s will appear in your workspace.

Downloading Selected Files from a Folder

This procedure describes how you can import a partial project, ignoring any files that you don't need and keeping them hidden on your workspace. For example, if you were importing a folder that contained image files that you didn't need, you could import all of the files, except the image files.

To import selected files:

1. Right-click on the new resource in PHP Explorer and from the right-click menu, select **Team**.
2. Click on **Addto.sharedignored** to open up a dialog box.
3. Select the files that do not need to be synchronized.
4. Click **Finish**.

The ignored files will be hidden in a folder on your workspace.

Uploading Resources to Your Server

This procedure describes how to upload resources to a remote server. This allows you to share your project to your remote server and all those who have access to it. Sharing your project allows you to edit and manage your project between different users and locations. For more information see Working with Remote Server.

Uploading a Project to a Remote Server

This procedure describes how to upload a project to a remote server. This allows you to share your project through your remote server.

Note:

Before you can share your project to a remote server, you must have already created a remote server connection.

Once you have defined a connection, you can share your local project to a connection.

Uploading a project to a Remote Server:

1. Right click on the shared project in the PHP Explorer view.
2. From the right click menu select **Team | Upload to Server.**

Your project will be uploaded to the remote server.

Once you have uploaded a project to the remote server, other users have the option to download the project from the server and edit it.

Importing Actions

This procedure describes how to import resources from another location. Before importing, you must have already created a remote server connection and synchronized the project that contains the resource/s.
To import resources:

1. Select the resource/s in the Synchronize View.
2. Right-click on the resource and from the right-click menu, select Import.

The resource/s will appear in your workspace.

You can also import resources via PHP Explorer:

1. Select the resource/s in PHP Explorer.
2. Right-click on the resource and from the right-click menu, select Import.

The resource/s will appear in your workspace.

Working with Mylyn Integration

The Task List contains two types of tasks:"Local Tasks" and shared "repository tasks" that are stored in a task repository such as Bugzilla or Jira. See how to create new tasks. Local tasks are typically contained in categories, which you can create by right-clicking on the task list and selecting **New** | **Category**. Repository tasks are contained in special categories that represent queries.

At the top of the Task List, you will find the following buttons and features:

- New Task - Create a new local or repository task.
- Synchronize - Update repository tasks with changes from the server.
- Task Presentation - Toggle between Scheduled and Categorized presentations.
- Focus on Workweek - See only tasks scheduled for this week.
- Find - search for a task by typing in words from the task summary
- Working set indicator - Indicates the currently active working set. Use the black arrow on the left to change the working set.
- Current task indicator - Indicates the currently active task. Use the black arrow on the left to re-activate a recently active task.

Task List Presentation

The task list supports several ways to present tasks. You can toggle between the following modes by using the "Task Presentation" button in the toolbar. Categorized - View tasks grouped by their category Scheduled - View tasks grouped by the "scheduled date" .

Icon Legend and Color Coding

See the legend below to interpret the icons and color coding in the task list. You can view this legend by selecting "Show UI Legend" from the menu that appears when you click the white down arrow next to the minimize button in the top right corner of the Task List view.

Creating new Tasks

You can create new tasks by clicking on the "New Task" button in the Task List's toolbar. This will open the "New Task" dialog and prompt you to select a repository. There are two main types of tasks:

- Local tasks
- Repository tasks

Local Tasks

You can use local tasks if you do not have a shared task repository or if you would like to create a private personal task that is local to your workspace.

To create a local task:
1. Select Local Task | **Finish** from the **New Task** dialog. You can then provide the following details about the task:

- Task Description - Your task is called New Task by default. Replace this with a brief description of your task.
- Priority - Set the priority of your task. This will affect the tasks' icon and order in the task list.
- Status - Set your task to "complete" or "incomplete". In your task list, completed tasks have a strike-through font and will appear lower in the list.
- URL - You can associate a URL with this task.
- "Retrieve Task Description from URL" button - Set the task description to the title of the associated URL (page)
- "Open with Web Browser" button - Open the URL in the integrated web browser
- Scheduled For - Set the date when you will work on this task. Tasks scheduled for today or a date in the past will appear in blue in your task list. Tasks scheduled for future days will appear in black. If your task list is in focused mode, only tasks for the current week will be visible (unless they have unread changes).
- Due - Set the date when your task must be completed. Overdue tasks and tasks due today will appear in red in your task list.
- Estimated Hours - Estimate the number of hours it will take to complete this task.
- Active - Displays the total time that you have worked on this task. Time is only recorded when this task is active and you are actively interacting with the system.
- Notes - Record your personal notes about this task.

Repository Tasks

You can create a new repository task when you would like to share information about the task with your team using a task repository such as Bugzilla or JIRA. To create a new repository task, click on the "New Task" button in the Task List's toolbar. You can then select the repository where you would like to create a task. If you don't see your team's task repository, you will need to configure it in the task repositories view. Once you have selected a repository, click "Next". If you are connecting to a Bugzilla repository, select a **Product** as a top-level category for your task and click "Finish". A new task editor will appear.

If you are using Bugzilla, you can enter the following required information:

- Description - Enter a brief task description in the text box at the top (this box does not have a label).
- Component - Specify a "Component" to further categorize this task within the previously selected "Product".
- Description - Describe the task in detail. Optional
- You can specify additional information about your tasks in the "Attributes" section.
- Personal Planning - You can enter information in this section that will be local to your workspace and not available on your team's task repository. See "Local Tasks" for more information about the personal planning fields.
- Assigned to - Specify who should work on the task. Type the first several characters of the person's email address, and then press ctrl+space to select the address from a list. A task can be assigned to only one person at a time.
- Add CC - Add the addresses of people who should be notified of changes to this task. You can add multiple addresses, separated by a comma, e.g. (mik.kersten@tasktop.com, steffen.pingel@tasktop.com).

When finished, click **Submit** to add your new task to your team's shared task repository.

Context

The context tab allows you to manage the context of resources associated with the task. You can view the context tab by selecting it in the lower left of the editor window.

Elements

This section lists the resources that are part of the tasks' context. Because the number of elements may be large, you can adjust the level of detail using the slider at the top of the Actions section. Sliding the control all the way to the left will show you all elements in your task context. As you slide it to the right, only the elements with a high level of interest will be displayed. You can manually remove elements from your task context by doing the following:
1. Right-Click **Remove From Context**.
You may choose to view all elements and exclude irrelevant items in this way before attaching the context to the task so that others can download it.

Actions

Element Detail Slider - Adjusts the minimum level of interest required for an element to be displayed in the Elements section.

- Attach Context - Attaches the context to the task so that it is available for download from the shared task repository. The context consists of the elements shown on the right.
- Retrieve Context - Replaces the current task context with one that is attached to the task in the shared task repository.
- Copy Context to... - Copy the task context to another task. That task will then have the same context as the current task.
- Clear Context - Removes all context information from the task.

Planning

Use the planning tab to access local information about the task that is private to your workspace. You can view the planning tab by selecting it in the lower left of the editor window. This tab contains a large area where you can enter personal notes about the task. See the local task section for more information about fields in the Personal Planning section.

Task-Focused Interface

The task-focused interface is oriented around tasks and offers several ways to focus the interface on only what is relevant for the currently active task. You can focus navigator views (e.g. Package Explorer, Project Explorer, Navigator) by toggling the "Focus on Active Task" button in the toolbar. When focused, the view will show only the resources that are "interesting" for the currently active task.

Alt+Click Navigation

To navigate to a new resource that is not a part of the active task's context, you can toggle "Focus on Active Task" off, browse to the resource, and then click "Focus on Active Task" again to see only relevant resources. A more efficient way to add new resources is to use Alt+Click navigation (Clicking the mouse while holding the Alt key). When a view is in Focused mode, you can Alt+Click a node to temporarily show all of its children. Once an element that was previously not interesting is selected with the mouse, it becomes interesting the other child elements will disappear. The clicked element is now a part of the task's context. Alt can be held down while clicking to drill down from a top-level element to a deeply nested element that is to be added to the task context. Multiple Alt+Click are supported so that you can add several elements to the task context. As soon as a normal click is made, uninteresting elements will disappear. Ctrl+Click (i.e. disjoint selections, use Command key on Mac) are also supported and will cause each element clicked to become interesting. The first normal click will cause uninteresting elements to disappear. Note that Ctrl+click elements will become interesting (turn from gray to black) but only the most recently-clicked one will be selected while Alt is held down. Focusing Editors

Some editors such as the Java editor support focusing. Clicking the Focus button in the toolbar will fold all declarations that are not part of the active task context.

Task-focused Ordering

When a task is active, elements that are interesting are displayed more prominently. For example, when you open the Java Open Type dialog (Ctrl+Shift+T), types that are interesting for the active task are shown first. Similarly, when you use ctrl+space to autocomplete a method name in a Java source file, methods that are in the task context are displayed at the top. Working Set Integration
When Focus is applied to a navigator view, the working sets filter for that navigator view will be disabled. This ensures that you see all interesting elements when working on a task that spans working sets. To enforce visibility of only elements within one working set, do the following: Set the view to show working sets as top-level elements. Use the Go Into action on the popup menu of the working set node in the view to scope the view down to just the working set. Open Task dialog
An Open Type style dialog is available for opening tasks (Ctrl+F12) and for activating tasks (Ctrl+F9). The list is initially populated by recently active tasks. The active task can also be deactivated via Ctrl+Shift+F9. This can be used as a keyboard-only alternative for multi-tasking without the Task List view visible. These actions appear in the Navigate menu.

Task Hyperlinking

In the task editor, comments that include text of the form bug#123 or task#123 or bug 123 will be hyperlinked. Ctrl+clicking on this text will open the task or bug in the rich task editor. To support hyperlinks within other text editors such as code or .txt files, the project that contains the file must be associated with a particular task repository. This is configured by right-clicking on the project and navigating to "Properties" > "Task Repository" and selecting the task repository used when working with this project. Reporting Bugs from the Error Log
Bugs can created directly from events in the Error Log view. This will create a new repository task editor with the summary and description populated with the error event's details. If the Connector you are using does not have a rich editor, the event details will be placed into the clipboard so that you can paste them into the web-based editor that will be opened automatically.

Developing with JavaScript

A range of JavaScript features are available within PHP files and projects, as well as in standalone JavaScript files.

The following tasks describe how to:

- Enable JavaScript Support in PHP Projects
- Set the JavaScript Build Path
- View JavaScript Elements in the Outline View
- Use and Configure JavaScript Content Assist

Note:

In order for Content Assist options to include elements from JavaScript libraries and referenced files and folders from outside of your project, you must first enable JavaScript Support in your project, and then set the JavaScript Build Path.

- Use and Configure JavaScript Mark Occurrences
- Use and Configure JavaScript Syntax Coloring
- Open JavaScript Types
- Set Up and Use Dojo Integration

Enabling JavaScript Support in PHP Projects

Enabling JavaScript support in PHP projects allows JavaScript libraries and external files to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

Note:

Once JavaScript support has been enabled for a project, you should set the project's JavaScript Build Path in order for the required resources to be made available to the project.

These procedures describe how to enable JavaScript support for new PHP projects, add support to existing projects, or disable JavaScript support.

Enabling JavaScript Support for New PHP Projects

To enable JavaScript support in new PHP Projects:

1. Go to File Menu and select **New | PHP Project**.
   -Or- In PHP Explorer view, right-click and select **New | PHP Project**.
   The new PHP Project wizard will launch.
2. Enter the required information in the various fields.
3. To enable JavaScript support, mark the 'Enable JavaScript support for this project' checkbox.

4. Click Finish.

A new PHP Project will be created with full JavaScript support.

Note:

You must now set the project's JavaScript Build Path in order for the required JavaScript libraries and files to be accessed by the project.

Enabling JavaScript Support for Existing PHP Projects

JavaScript libraries and features can be added to existing PHP projects in your workspace.

To enable JavaScript Support for existing PHP Projects:

In PHP Explorer view, right-click the project for which you want to enable JavaScript support and select Configure | Add Java Script Support.

JavaScript support will be enabled for the project.

Note:

You must now set the project's JavaScript Build Path in order for the required JavaScript libraries and files to be accessed by the project.

Removing JavaScript Support

If you are not using JavaScript libraries or files in your project, you can remove JavaScript support for that project.

To remove JavaScript Support for existing PHP Projects:

In PHP Explorer view, right-click the project for which you want to enable JavaScript support and select Configure | Remove JavaScript Support.

JavaScript support will be removed from the project and no JavaScript libraries or external files will be available to the project.

Setting the JavaScript Build Path

The JavaScript build process scans all files, folders, projects and libraries that are on the project's JavaScript Build Path so that resources which are referenced in the project can be made available for Content Assist options and Refactoring operations. The required resources must therefore be included in the JavaScript Build Path.

JavaScript's Build Path allows you to select JavaScript resources to include/exclude from this process. Rather than automatically scanning all referenced projects and libraries, configuring the JavaScript Build Path allows you to select which resources will be scanned, and so can significantly speed up the build process.

Note:

JavaScript Support must be enabled for the project before you can set the JavaScript Build Path.

The JavaScript Build Path is determined by the resources included in the project's JavaScript Libraries properties page.

To configure the project's JavaScript Build Path:

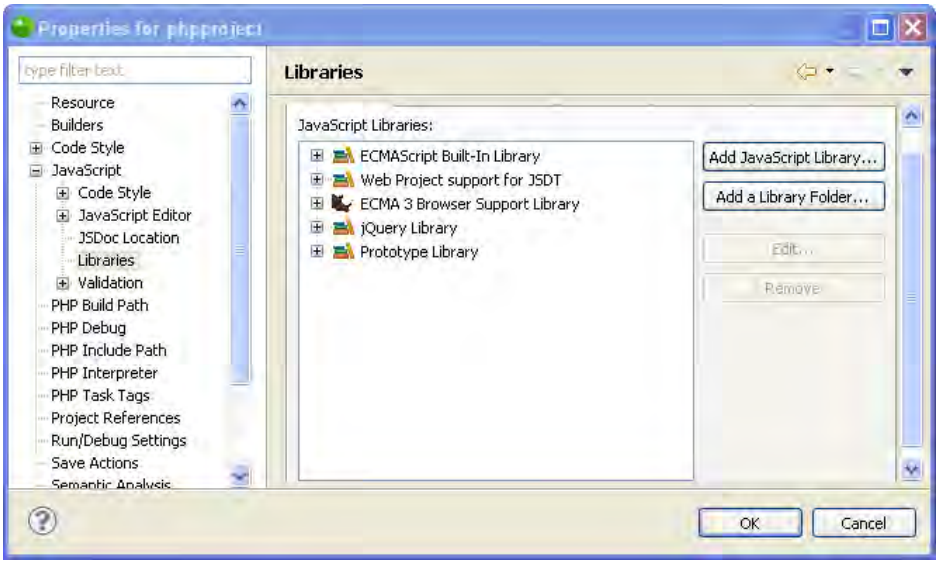
1. In PHP Explorer view, right-click the required project and select Properties | JavaScript | JavaScript Libraries.
   In this properties page you can add the following elements to the project's JavaScript Build Path:
   - Libraries - Built-in and user defined libraries contain pre-written collections of JavaScript source files that have prototyped object/class definitions and JsDoc.
   - Global Super Types - Each object/field/function in the added libraries will be added to the project's Global Scope. Every JavaScript file in the project will then have access to these objects/fields/functions.
   - Source - Other folders/files/variables from your file system or linked from an external source.
   - Projects - Other projects in the workspace or on your file system

Libraries Tab

If your project references a JavaScript library, it should be added to this list so that it's elements can be available for Content Assist options.
Through the Libraries tab you can add Zend Studio's built-in libraries, or create and add your own User Libraries.

Note:

The default Runtime Libraries (Script Language Libraries, Web Project support for JSDT and ECMA 3 Browser Support Library) are added to the JavaScript Build Path automatically.

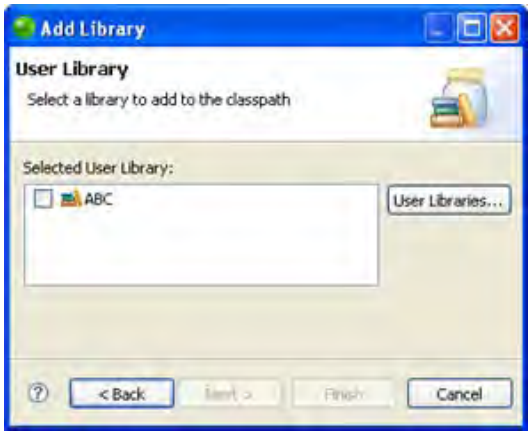


To add a built-in library to the JavaScript Build Path:

1. Click 'Add a Runtime Library...'.
   The Add Library dialog will appear.
2. Select the required library and click Next and Finish.

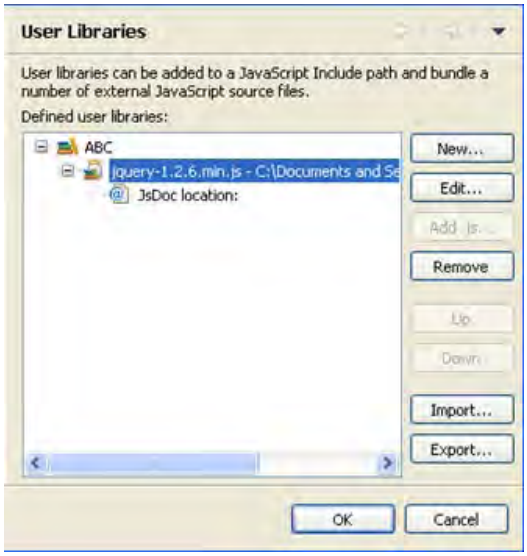
To add a User Library to the JavaScript Build Path:

1. Click 'Add a Runtime Library...'.
   The Add Library dialog will appear.
2. Select 'User Library' and click Next.
   A list of previously configured User Libraries will appear.



3. If you have not yet configured your User Libraries:
   i. Click the User Libraries button to be taken to the JavaScript User Libraries preferences page (also accessible from Window | Preferences | JavaScript | Include Path | User Libraries).
   ii. Click New.
   The New User Library dialog will open.

   iii. Enter a name for the library you would like to add and click OK.
   The library name will be added to the User Libraries list. This is just a placeholder name to which you will add the required files in order to create your own library.
   iv. Select the name from the list and click 'Add .js.'
   v. In the .js File Selection dialog, browse to the required JavaScript files on your file system and click Open.
   vi. The JavaScript files will be added to your User Library.



vii. Click OK to return to the User Library selection list.

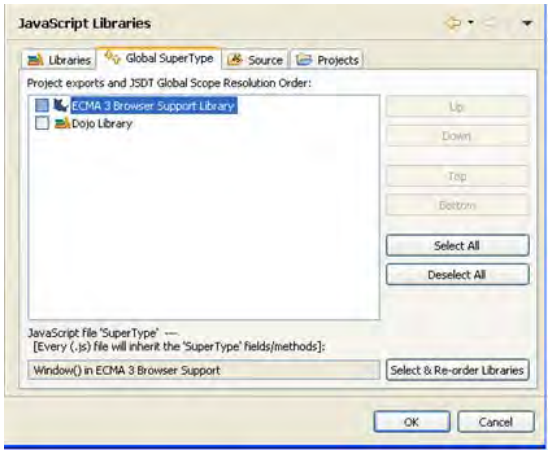
4. Mark the checkbox of the User Library you would like to add to the project's Build Path and click Finish.
   The User Library will be added to the project's JavaScript Libraries Build Path list.

To add a library folder from your Workspace to the JavaScript Build Path:

1. Click the 'Add a Library Folder' button.
   The library folder selection dialog will display a list of available folders.
2. Select the required folder(s) and click OK.

Global SuperType Tab

The project's Global Scope contains all the objects, fields and functions contained in the libraries which have been added to the JavaScript Build Path. Elements in the Global SuperType will be available to all PHP and JavaScript files within the project.



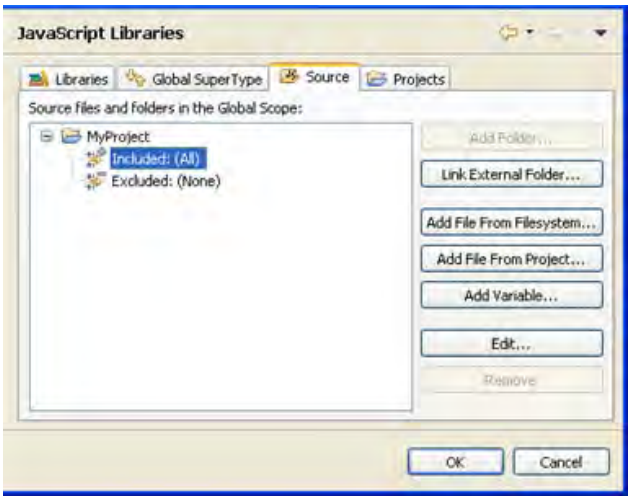
In addition, the Global SuperType tab allows you to select a 'SuperType' class. All JavaScript elements which are not defined in their own class will automatically be added to this class. This will allow you to select these elements from the Content Assist list according to the SuperType Class name you selected.

To configure the SuperType class:

1. Click the 'Select & Re-order Libraries' button.
   The JavaScript Sourcefile SuperType Selection dialog will open.
2. Select the class name for the SuperType class and click OK.
   All elements which are not defined in a class will be added to the selected class. Entering the class name will display these elements in the Content Assist list.

Source Tab

The Source tab allows you to include/exclude specific files, folders and variables from the JavaScript Build Path.



To add a folder from your Workspace:

1. Click Add folder.
   The Source Folder Selection dialog will display.
2. Select the required folders and click OK.

To add an external folder:

1. Click Add External Folder.
   The Link Source dialog is displayed.
2. Click Browse to select a folder from your file system and click OK.

To add a JavaScript file from your filesystem:

1. Click Add File from Filesystem.
2. Select a JavaScript file and click Open.

To add JavaScript files from your Workspace:

1. Click Add File From Project.
2. Select the required file(s) and click OK.

Using a Classpath Variable allows you to point to a JavaScript library without having to specify it's local file system location. This is important when sharing resources in a team.

To add a Variable Classpath Entry:

1. Click Add Variable.
2. The list of Configured Classpath Variables will be displayed.
   If you have not configured any Classpath Variables:
   i. Click the Configure Variables button to open the JavaScript Include Path Variables preferences page (also accessible from Window | Preferences | JavaScript | Include Path | Include Path Variables).
   ii. Click New.
   The New Variable Entry dialog will appear.
   iii. Enter the Name and Path of the variable (click the File or Folder buttons to browse to the path).
   iv. Click OK.
   The Variable will be added to the Include Path Variables list.
   v. Click OK to return to the Variable Classpath Entry selection list.
4. Select the variable you would like to add to the project's JavaScript Build Path and click OK.

Note:
The Variable points to a folder. To choose an archive inside the folder, click Extend.

5. Click OK.

You can choose to include or exclude specific resources or file name patterns from within source folders which you have added.

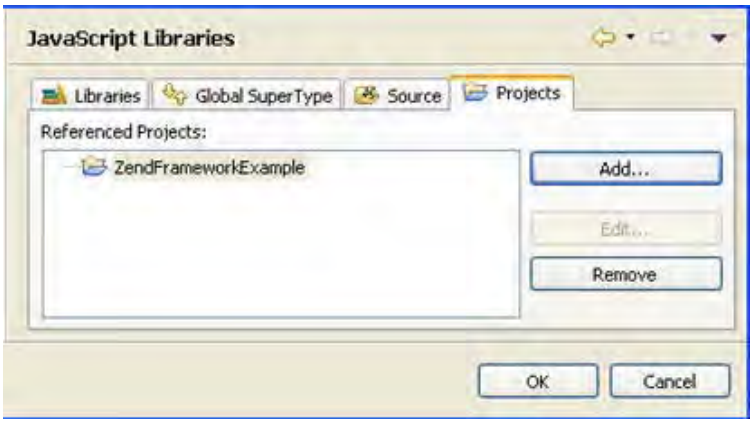
To include/exclude specific resources from the JavaScript Build Path:

1. Expand the node next to the folder from which you would like to include/exclude resources.
   A list of the resources which are included and excluded will be displayed (by default, all resources within the folder are included).
2. Select the Included or Excluded list and click Edit.
   The Inclusion and Exclusion patterns dialog is displayed.
3. Click Add next to the Inclusion or Exclusion patterns panes.
   The Add Inclusion/Exclusion Pattern dialog is displayed.
4. Enter or select the required resources / pattern to Include/Exclude and click OK.
5. Click Finish.
   All resources in the folder which match an Inclusion pattern but do not

match an Exclusion pattern will be added to the JavaScript Build Path.

Projects Tab

The Projects tab allows you to select projects from your Workspace to add to the JavaScript Build Path.



To add a project to the JavaScript Build Path:

1. Click the 'Add..' button.
   The Required Project Selection dialog is displayed.
2. Select the required projects and click OK.
   The selected projects will be added to the JavaScript Build Path.

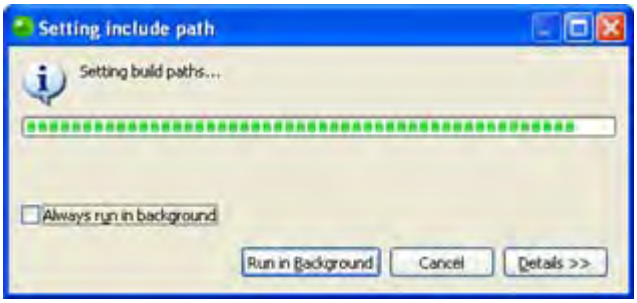
Note:

During the Build process, the referenced projects will be built before the project itself.

Saving your changes

Once you have made all necessary changes to your JavaScript Build Path,
click OK.

Your project will be rebuilt to reflect the changes, according to the settings you configured.

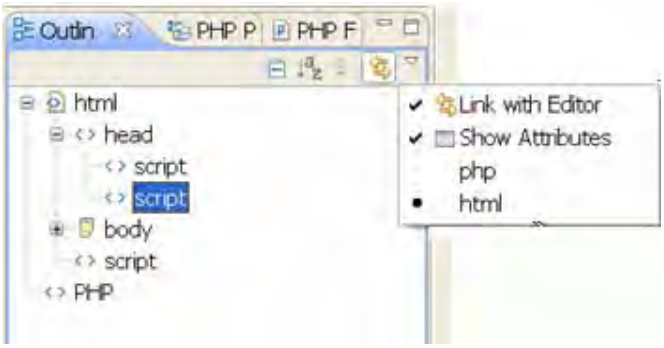


Viewing JavaScript Elements in the Outline View

This procedure describes how to view JavaScript objects and elements in the Outline view.

To view your JavaScript objects in the Outline view:

1. Go to the Outline view.
   If it is not displayed, go to Window | Show View | Outline.
2. If your JavaScript objects are contained within a PHP file, click the Menu arrow on the Outline view's toolbar and select html.
   HTML and JavaScript objects contained within the file will be displayed in a tree view.



3. Double-Clicking the <Script> node in the outline view will select the entire <script> element in the Editor.

Using JavaScript Content Assist

These procedures describe how to enable and configure JavaScript Content Assist options.

Note:

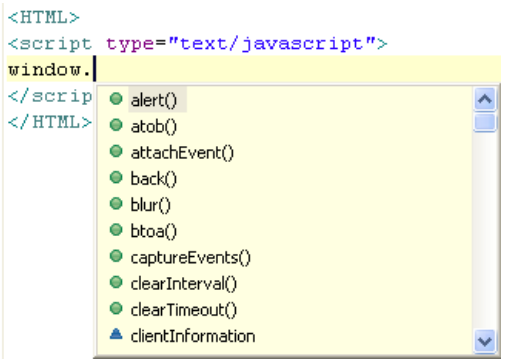
In order for JavaScript elements from libraries and files outside of the project to be made available for use in the Content Assist list, you must enable JavaScript Support for the project and Set the JavaScript Build Path. To access Dojo toolkit content assist options, you must set up Dojo integration in the project while configuring the Build Path.

To access JavaScript Content Assist options:

1. Type the relevant HTML and JavaScript tags:

```
<HTML>
<script type="text/javascript">

</script>
</HTML>
```

2. JavaScript Content Assist options will now be available.
3. Type the first few letters of the required element.
   The Content Assist list will be displayed (if the Content Assist list is not automatically displayed, press Ctrl + Space or go to your JavaScript Content Assist preferences page to configure your auto activation preferences).
4. Select the relevant option from the Content Assist window by double-clicking or pressing Enter.
5. If you selected a JavaScript class, type a period "." after the name of the class to display a Content Assist window with the classes' relevant functions and methods.



6. Select the required option to complete your JavaScript code.

JavaScript Content Assist Configuration

JavaScript Content Assist options can be configured from the JavaScript Content Assist preferences page.

To configure JavaScript Content Assist options:

1. Go to the JavaScript Content Assist preferences page, accessed from **Window | Preferences | Web | JavaScript | Editor | Content Assist**.

2. Configure the following options, according to your preferences:

Insertion

- Completion Inserts/Completion Overwrites - Select whether choosing an item from the Content Assist list will cause new code to be entered or existing code to be overwritten.
- Insert single proposals automatically -If enabled, the content assist suggestion will be inserted automatically when only one content assist option exists
- Insert common prefixes automatically - If enabled, Content Assist will

automatically insert the common prefix of all possible completions similar to Unix shell expansion. This can be used repeatedly, even while the Content Assist window is being displayed.

- Fill argument names on completion - If enabled, Content Assist will add arguments when completing a method.
- Guess filled function arguments - If enabled, Content Assist will fill the arguments with the best matching function, according to the context.

Sorting and Filtering

- Sort proposals - Select how the proposals should be sorted in the Content Assist list.
- Hide proposals not visible in the invocation context - If enabled, the Java element proposals are limited by the rules of visibility. For example, private field proposals of other classes would not be displayed.
- Show camel case matches - If enabled, camel case matches are displayed (e.g. NPE is expanded to NullPointerException).
- Hide forbidden references - If enabled, references to JavaScript elements forbidden by access rules are not displayed.
- Hide discouraged references - If enabled, references to JavaScript elements discouraged by access rules are not displayed.
- Hide deprecated references - If enabled, references to deprecated JavaScript elements are not displayed.

Auto-activation

- Enable auto activation - If enabled, the Content Assist list will automatically be displayed when the first letters of an element are typed.

Note:
If this is unmarked, you can display the Content Assist list by pressing Ctrl+Space.

- Auto activation delay - Determines the delay before the Content Assist box is automatically displayed.
- Auto-activation triggers for JavaScript - Sets the characters that determine JavaScript context.
- Auto-activation triggers for JSDoc: Sets the characters that determine JSDoc context.

3. Click Apply to apply your settings.

Using JavaScript Syntax Coloring

JavaScript Syntax Coloring can be applied to JavaScript code within a PHP file or a JavaScript file.

To enable JavaScript syntax coloring:

Enter the relevant JavaScript tags:

```
<HTML>
<script type="text/javascript">

</script>
</HTML>
```

Any code enclosed in the JavaScript tags will have Syntax Coloring applied to it, according to the configuration in the JavaScript Syntax Coloring Preferences page

```
<HTML>
<script type="text/javascript">
/**
 * This is a multiple-
 * line comment
 */
var index = 0;
var arr = [];

function push(elem) {
    // This comment may span only this line
    arr[index++] = elem;
}
</script>
</HTML>
```

JavaScript Syntax Coloring Configuration

JavaScript Syntax Coloring options can be configured from the JavaScript Syntax Coloring preferences page.

To configure JavaScript Syntax Coloring options:

1. Go to the JavaScript Syntax Coloring preferences page, accessed from **Window | Preferences | Web | JavaScript | Editor | Syntax Coloring**.

2. Select the required item from the Syntax element list.
   To enable Syntax Coloring for the element, ensure the "Enable" checkbox is marked.
3. Select what formatting, if any, you would like to apply to the text (Bold, Italic, Strikethrough, Underline).

Note:
The Sample text box displays a preview of the different elements.

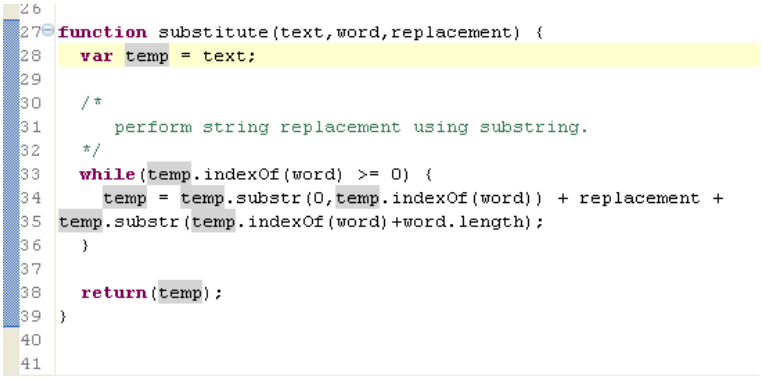
5. Click **Apply** to apply your settings.

Using JavaScript Mark Occurrences

The JavaScript Mark Occurrences feature allows you to see where a variable, method or type is referenced within a JavaScript file.

To use Mark Occurrences:

Stand on a variable, method or type in your file.

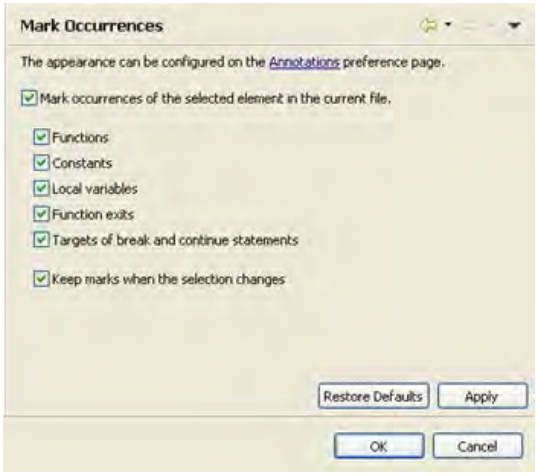


```
26
27 function substitute(text,word,replacement) {
28   var temp = text;
29
30   /*
31      perform string replacement using substring.
32   */
33   while(temp.indexOf(word) >= 0) {
34     temp = temp.substr(0,temp.indexOf(word)) + replacement +
35 temp.substr(temp.indexOf(word)+word.length);
36   }
37
38   return(temp);
39 }
40
41
```

All instances where the element is referenced within the file will be highlighted.

To configure Mark Occurrences settings:

1. To toggle mark occurrences, click the Toggle Mark Occurrences button on the toolbar -or- press Alt+Shift+O.
2. To configure further Mark Occurences preferences, go to the JavaScript Mark Occurrences preferences page, accessed from **Window | Preferences | Web | JavaScript | Editor | Mark Occurrences**.



3. Mark the "Mark occurrences of the selected element" checkbox to enable the Mark Occurrences feature.
4. Select which element's occurrences will be marked by marking the checkboxes next to the required elements.
5. Mark the "Keep marks when the selection changes"' checkbox for marks to continue to be displayed once the cursor has been moved from the selected element.
6. Click **Apply** to apply your changes.

Note:

The appearance of marked occurrences can be configured in the Annotations preferences page (**Window | Preferences | General | Editors | Text Editors | Annotations**).

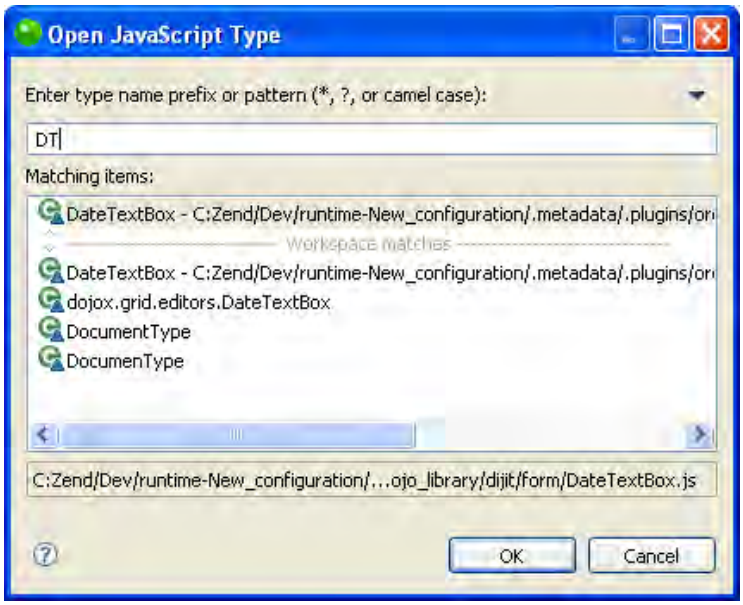
Opening JavaScript Types

The JavaScript Open Type functionality allows you to search for any JavaScript type in your workspace and opens an editor with the type's declaration.

To open a JavaScript Type:

1. Open the JavaScript Perspective by going to Window | Open Perspective | Other | Javascript.
2. Click the Open JavaScript Type icon on the toolbar - or - from the menu bar go to Navigate | Open JavaScript type...
   The Open JavaScript Type dialog will open.
3. Enter the first few letters of the class which you would like to open. (You can also use wildcards such as * or ?, or camel case.)
   The list of available types will be filtered to match the expression you are entering.

Note:
If you have previously opened a Type which matches your expression, this will be displayed first above a separator line.



4. Select the required class and click OK.

An editor will open with the required class's declaration.

Setting Up and Using Dojo Integration

Zend Studio's Dojo Integration functionality allows you to add the Dojo toolkit library to your project's Build Path so that its functions, classes and elements will be available for operations such as code completion and hover help.
See http://dojotoolkit.org/ for more information on the Dojo toolkit.

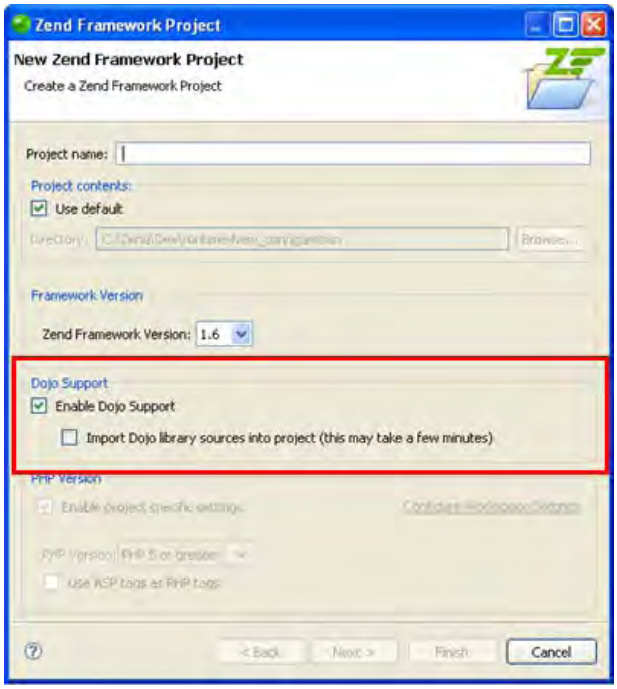
Setting Up Dojo Integration in PHP Projects

To enable Dojo integration you must first enable JavaScript Support in your PHP Project.
You then need to add the Dojo library to your project's Build Path by following the instructions under the Adding Built-in Libraries section of Setting the JavaScript Build Path topic, and selecting the Dojo library.

Setting Up Dojo Integration in Zend Framework Projects

The Dojo library can also be added to Zend Framework projects through the New Zend Framework Project wizard.
When creating a Zend Framework project, mark the 'Enable Dojo Support' checkbox in the Dojo Support category. This will create a link to the external Dojo toolkit library, allowing for Dojo development functionality.

Note:
This option will only be available if Zend Framework version 1.6 was selected.



In addition, you can select to import the JavaScript library into your project. The link to the external library will still be maintained, so you do not have to add this Dojo library to your JavaScript Build Path (this will speed up the JavaScript build process).

Note:
The Dojo library may take a couple of minutes to import. If you are using Dojo for development purposes only, we recommend that you do not import it into your project.

If you did not enable Dojo support for the project during its creation, you can do so by adding the Dojo for Zend Framework library to the project's build path.
See the Adding Built-in Libraries section of Setting the JavaScript Build Path topic for more information.

Using Dojo Integration

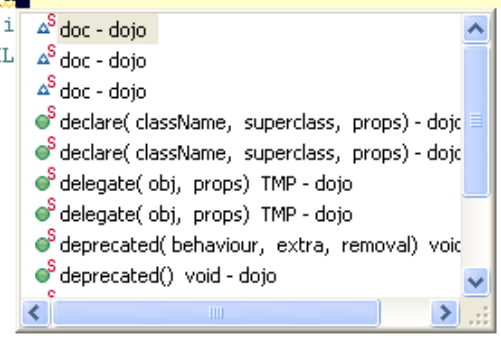
Once Dojo libraries are included in your project's JavaScript Build Path, Dojo code completion options will be available to files within that project.

The following Dojo elements will be available in the content assist list:

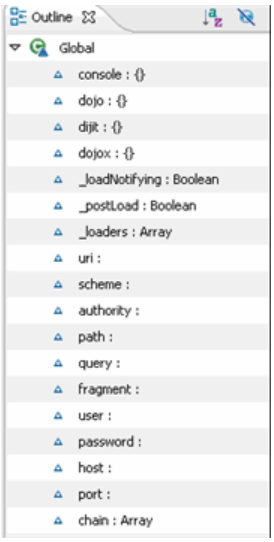
- Dojo classes (including namespaces)
- Dojo object methods
- Dojo object properties
- Dojo global variables (e.g. dojo)
- Classes in dojo.require('r;')

Example:

```
<HTML>
<script type="text/javascript">
dojo.d
</scri
</HTML
```


doc - dojo
doc - dojo
doc - dojo
declare(className, superclass, props) - dojc
declare(className, superclass, props) - dojc
delegate(obj, props) TMP - dojo
delegate(obj, props) TMP - dojo
deprecated(behaviour, extra, removal) voic
deprecated() void - dojo

Dojo elements will also be displayed in the Outline view:



Dojo requireModule

The dojo() view helper is intended to simplify setting up the Dojo environment and can be accessed from within Zend View files.

One of the methods available from the Dojo view helper is requireModule, which sets up a require method. Within Zend View files, content assist options will be available to this method:

Example:

See http://framework.zend.com/manual/en/zend.dojo.view.html for more information.

Reference

PHP Perspectives and Views	Keymap
PHP Perspective Menus	Useful Links
PHP Perspective Main Toolbar	Contribute to the Documentation
PHP Preferences	Support
PHP Project Properties	Registering Your License
PHP Icons	

PHP Perspectives and Views

Zend Studio incorporates a number of Perspectives and Views for managing all aspects of your PHP code, files, project and application creation.

The following PHP perspectives are used for developing PHP:

- PHP Perspective
- PHP Debug Perspective
- PHP Profile Perspective
- Zend Server Code Tracer Perspective

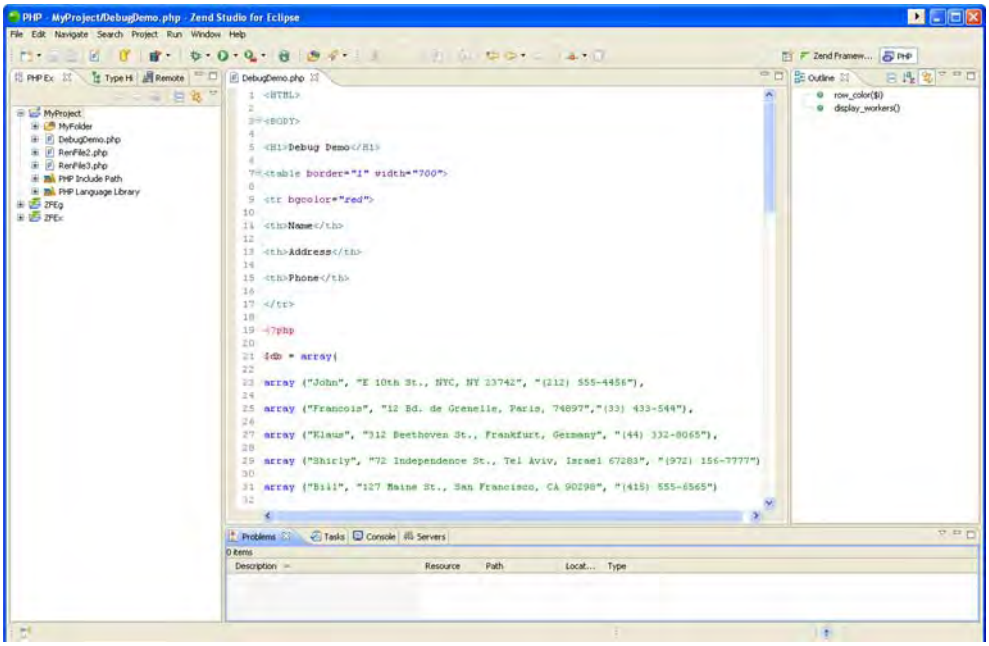
See the Workbench User Guide for more on Perspectives and Views.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

PHP Perspective

The PHP Perspective is Zend Studio's default perspective. It incorporates all Zend Studio's PHP project/file creation, inspection and editing functionality.



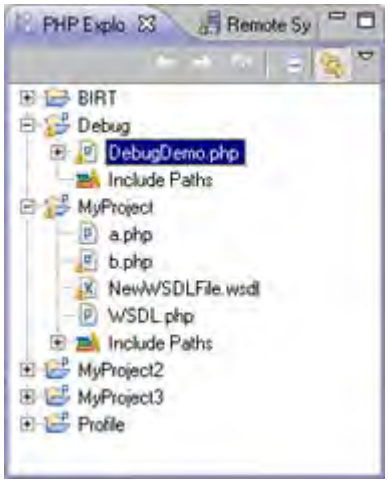
The PHP Perspective contains the following views:

- PHP Explorer View
- Outline View
- Type Hierarchy View
- Problems View (External Link)
- Tasks View (External Link)
- Console View (External Link)
- Servers View

PHP Explorer View

The PHP Explorer view is an internal file system browser, allowing you to view all PHP projects and files in your Workspace. It shows the PHP element hierarchy of PHP projects in the Workbench and provides you with a PHP-specific view of your available resources. Within each project, source folders and referenced libraries are shown in the tree. In addition, this view shows all PHP code elements (classes, functions, variables, etc.). Clicking an element or declaration will cause the corresponding code to be displayed in the PHP editor.

See PHP Icons for a description of the icons displayed in the PHP Explorer view.



PHP Explorer View Toolbar Commands

Icon	Name	Description
	Back/Forward	Scrolls backwards and forwards through your recently navigated resources. These icons will only be active if the 'Go into the selected element' option is selected in the PHP Preferences Page (available from Window | Preferences | PHP).
	Up	Navigates up one level. This icon will only be active if the 'Go into the selected element' option is selected in the PHP Preferences Page (available from Window | Preferences | PHP).
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

PHP Explorer View Menu Commands

The view's menu can be accessed through the view menu icon .

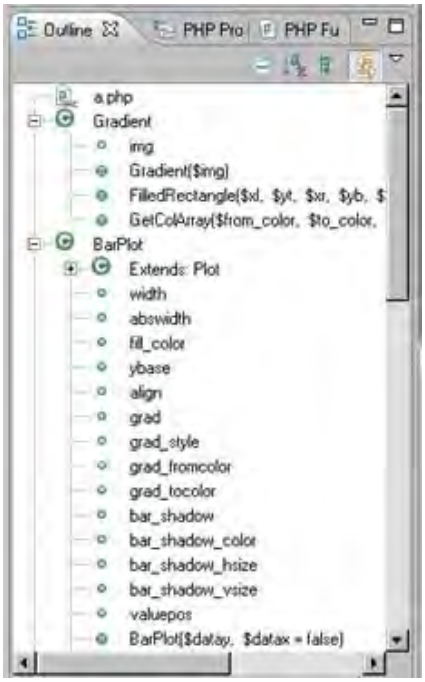
Icon	Name	Description
	Show	Select to view your projects grouped by Project or Working Set.
	Select Working Set	If Show Projects was selected (above), allows you to select a specific Working Set to view. See PHP Working Sets for more information.
	Deselect Working Set	Deselects the Working Set (if selected).
	Edit Active Working Set	Allows you to edit the selected Working Set. See PHP Working Sets for more information.
	Filters..	Opens the PHP Elements filters dialog which allows you to select which elements will be excluded from being displayed in the view. You select to exclude elements according to name or type.
	Group by namespaces	Sorts elements by namespaces (for projects using PHP 5.3 only).
	Link With Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

Note:

The PHP Explorer View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | PHP Explorer.

Outline View

The Outline view displays all PHP elements and element types in the current active file. The elements are grouped according to type and are displayed in a tree-like browser.
See PHP Icons for a description of the icons displayed in the Outline view.



Features

- The Outline View is updated interactively according to changes made in the files.
- Each type of PHP element is represented by a unique icon:
  - Reserved PHP Words
  - Functions
  - Templates
  - Classes
  - Interfaces
  - Constants
  - Variables (public)
  - Namespaces (PHP 5.3 only)
  - Use Statements (PHP 5.3 only)
- The Outline view is integrated with the Editor. Therefore, if you select a PHP element in the view, the Editor will jump to the element's declaration in the file in which it is declared.

Note:

Toggle the link to Editor on/off using the Link with Editor button .

- The View enables you to add PHPdoc blocks and, if available, Getters and Setters to any PHP element:

To generate a PHP DocBlock or Getter and Setter:

1. Right-click the element in Outline view.
2. Select Source | Add PHP Doc -or- Generate Getters and Setters.

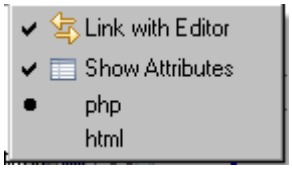
The relevant PHP DocBlock or Getter/Setter will be created above the code for the selected element in the edito

Outline View Toolbar Commands

Icon	Name	Description
	Collapse All	Collapses the list of elements
	Sort	Sorts the list alphabetically
	Show Groups	If selected, elements will be displayed in Groups (include files, constants, classes, functions)
	Link with Editor	If selected, elements will immediately be displayed in the editor when single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.

Outline View Menu Commands

The view's menu can be accessed through the view menu icon .



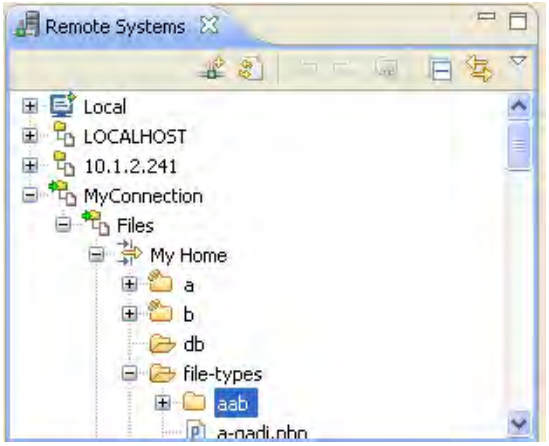
Icon	Name	Description
	Link with Editor	If selected, elements will immediately be displayed in the editor when single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.
	Show Attributes	If selected, element attributes will be displayed. These are defined by the element's PHP Doc Block.
php html	PHP/HTML selection	Toggles the view to display PHP or HTML elements.

Note:

The Outline View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | Outline.

Remote Systems View

The Remote Systems view helps you create, view and manage your connections to remote systems such as FTP and SFTP.



Remote Systems view Toolbar Commands

Icon	Name	Description
	Define a connection to remote system	Opens the 'New Connection' dialog.
	Refresh information of selected resource	Refreshes the connection information of selected resources.
	Back/Forward	Scrolls backwards and forwards through your projects.
	Up	Navigates up one level
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, files will immediately be displayed in the editor when selected. If unselected, files will be displayed in the editor when they are double-clicked.

Remote Systems View Menu Commands

The Remote Systems view menu can be accessed through the view menu icon .

Name	Description
New Connection...	Opens the 'New Connection' dialog.
Work with Profiles	Opens the Team profile view. See 'Remote System Explorer Profiles' in the RSE User Guide for more information. **Note:** Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).
Refresh All	Refreshes all connections.
Quality Connection Names	Displays the connection names.
Show Filter Pools	Displays filter pools. See 'Filters, filter pools, and filter pool references' in the RSE User Guide for more information. **Note:** Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).
Restore Previous State	Select this option to use locally cached information instead of connecting immediately if you are automatically opening the previously expanded connections when starting RSE.
Preferences	Opens the Remote Systems Preferences page.

Note:
The Remote Systems View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | Remote Systems | Remote Systems.

See the RSE User Guide for more on FTP/SFTP connectivity.

Note:

Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

Servers View

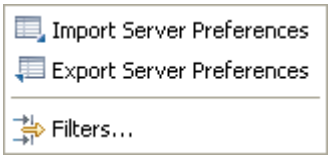
The Servers view displays a list of all servers configured in Zend Studio, and allows you to connect to and view any Zend Servers configured on your system.

Servers view Toolbar Commands

Icon	Name	Description
	Show Server Event List	Opens the Zend Server event list for the Zend Server selected in the Servers list (if applicable).
	Enable Tunnelling	Opens a tunneling connection to the server selected in the Servers list.
	Auto Detect Zend Server	Auto detects a Zend Server installed on the local machine.
	Collapse All	Collapses the list of servers.
	New Server...	Opens the PHP Server Creation dialog.

Outline View Menu Commands

The view's menu can be accessed through the view menu icon .



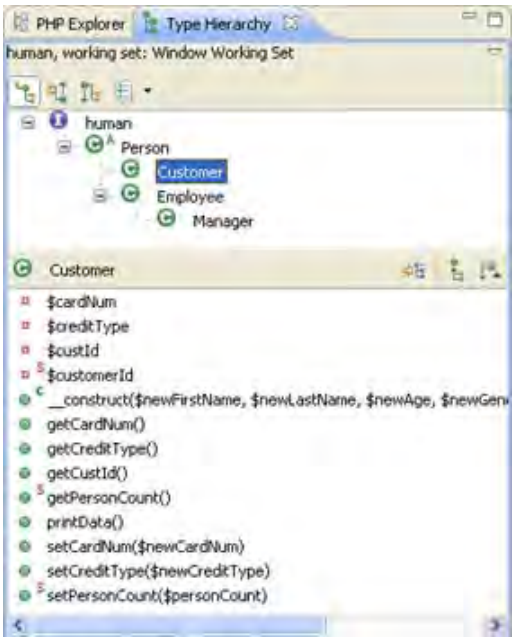
Icon	Name	Description
	Import Server Preferences	Launches the import server preferences dialog
	Export Server Preferences	Launches the export server preferences dialog
	Filters...	Launches the PHP Server Filters dialog which allows you to apply filters to the servers displayed in the Servers list.

Note:

The Servers View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | Zend Servers | Servers.

Type Hierarchy View

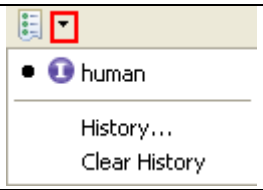
The Type Hierarchy view display the hierarchy for a given type (a class name, interface name or class methods, constants and fields.). This allows you to view an element's supertypes (types higher in the hierarchy) or subtypes (lower in the hierarchy) within a tree structure, providing you with an overview of your element's structure.
See Viewing Types in the Type Hierarchy View for information on how to open a type in the Type Hierarchy view.



The Type Hierarchy view consists of two panes:

- The Type Hierarchy Tree - Displays the type's supertypes and/or subtypes.
- Member list pane - Displays the type's members.

Type Hierarchy Tree Pane Toolbar Commands

Icon	Name	Description
	Show the Type Hierarchy	Displays the type in its full context (i.e., superclasses and subclasses) in the Type Hierarchy view.
	Show the Supertype Hierarchy	Displays the supertypes and the hierarchy of all implemented interfaces of the type. Note: The selected type is always at the top level, in the upper-left corner.
	Show the Subtype Hierarchy	Displays the subtypes of the selected class and/or all implementors of the interface in the Hierarchy view. Note: The selected type is always at the top level, in the upper-left corner.
	Previous Hierarchy Inputs	Displays a history of previously displayed type hierarchies.

Type Hierarchy Member List Pane Toolbar Commands

Icon	Name	Description
	Lock View and Show Members in Hierarchy	Only displays the members implementing the selected method. When the view is locked, the member list pane no longer tracks the selection in the Type Hierarchy Tree Pane.
	Show All Inherited Members	Shows or hides all methods and fields inherited by base classes. When this option is enabled, the name of the type that defines the method is appended to the method name.
	Sort Members by the Defining Type	Sorts the members according to the type in which they are defined.

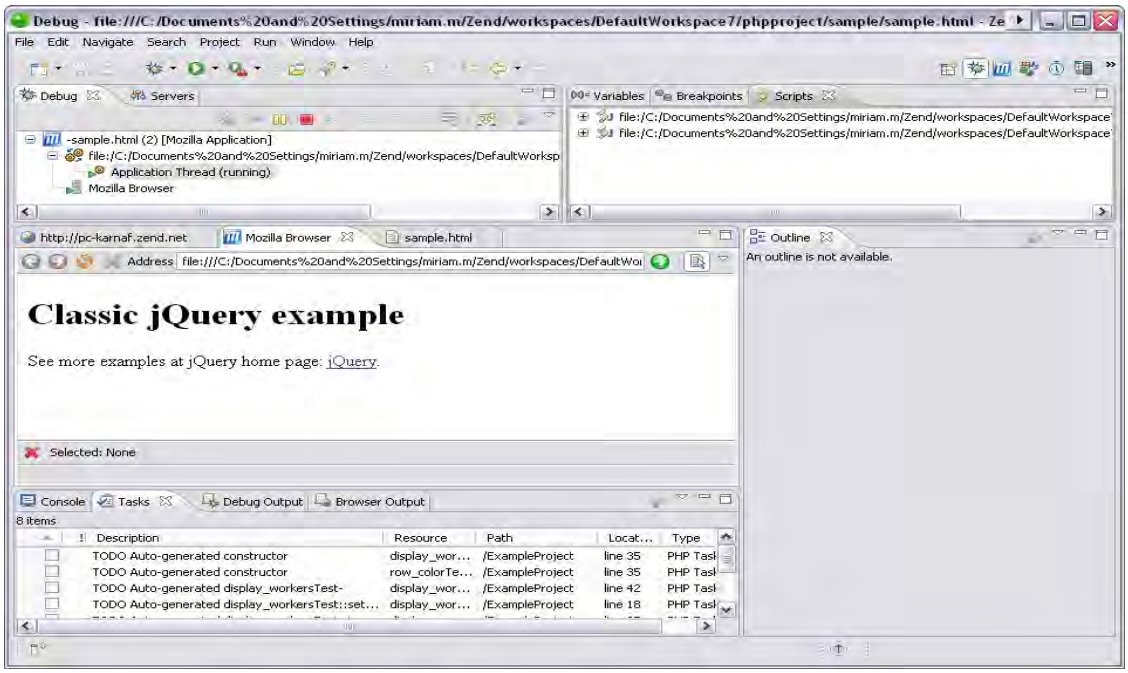
Type Hierarchy View Menu Commands

The view's menu can be accessed through the view menu icon .

Icon	Name	Description
	Show the Type / Supertype or Subtype Hierarchy	Selects whether to display the type, supertype or subtype hierarchy in the Type Hierarchy Tree Pane (see above).
	Select/Deselect/Edit Working Set	Allows you to view only types from within a specific Working Set. See PHP Working Sets for more information.
	Layout	Allows you to select the display of the panes within the Type Hierarchy view. The options are: ▪ Vertical View Orientation ▪ Horizontal View Orientation ▪ Automatic View Orientation ▪ Hierarchy View Only
	Show Qualified Type Names	Displays only qualified type names
	Link With Editor	If selected, elements will immediately be displayed in the editor when selected. If unselected, elements will be displayed in the editor when they are double-clicked.

PHP Debug Perspective

The PHP Debug Perspective can be launched automatically when a Debug session is run. It contains views which allow you to control and monitor the debugging process.


Debug - file:///C:/Documents%20and%20Settings/miriam.m/Zend/workspaces/DefaultWorkspace7/phpproject/sample/sample.html - Ze
File Edit Navigate Search Project Run Window Help
Debug Servers
Variables Breakpoints Scripts
-sample.html (2) [Mozilla Application]
Application Thread (running)
Mozilla Browser
http://pc-karnaf.zend.net
Mozilla Browser
sample.html
Address file:///C:/Documents%20and%20Settings/miriam.m/Zend/workspaces/DefaultWo
Classic jQuery example
See more examples at jQuery home page: jQuery.
Selected: None
Outline
An outline is not available.
Console Tasks Debug Output Browser Output
8 items
Description Resource Path Locat... Type
TODO Auto-generated constructor display_wor... /ExampleProject line 35 PHP Task
TODO Auto-generated constructor row_colorTe... /ExampleProject line 35 PHP Task
TODO Auto-generated display_workersTest- display_wor... /ExampleProject line 42 PHP Task
TODO Auto-generated display_workersTest::set... display_wor... /ExampleProject line 18 PHP Task

The PHP Debug Perspective contains the following views:

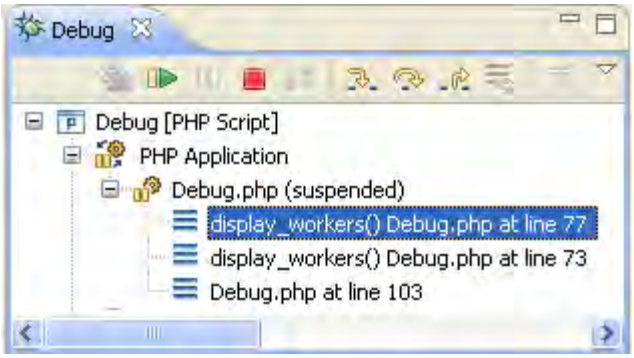
- Debug View - Here you can control (stop, pause, and resume) the debugging process. You can also decide whether to step into, step over or step return (step out of) certain functions.
- Variables View - Displays the various variables in your script.
- Breakpoints View - Displays the breakpoints you have entered
- Parameter Stack View - Displays the parameters through which functions are reached.
- Debug Output View - Displays the textual output of the script. This will be updated as the debugging process continues.
- Browser Output View - Displays the output of the script to a browser. This will be updated as the debugging process continues.
- Expressions View - Displays the progress of selected variables. The view will only be displayed if you have selected to watch a variable.
- Editor - Displays the code at the relevant sections, according to which line is selected in the Debug View window.
- Console View (External Link) - Displays any error and warning messages
- Tasks View (External Link) - Displays tasks that were added to your script (if applicable).

Note:

By default, a dialog will appear asking whether you want to open the Debug Perspective when a debugging session is run. To change this behavior, open the Perspectives Preferences dialog by going to Window | Preferences | Run/Debug | Perspectives and select Always, Never or Prompt in the 'Open the associated perspective when launching' category.

Debug View [PHP Debug Perspective]

The Debug view displays the stack trace and allows you to monitor and control the Debugging process.



Debug View Toolbar Commands

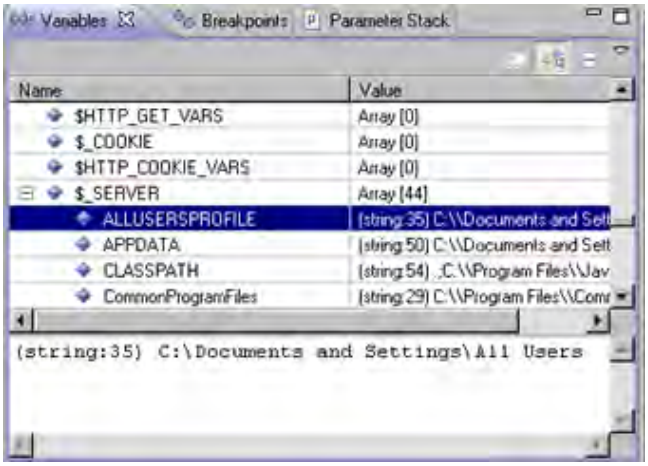
Icon	Name	Description
	Remove Terminated Launches	Remove any terminated debug sessions from the list.
	Resume	Continue the debugging process until the next breakpoint, or until the end of the debugging process.
	Terminate	Stop the debugging process.
	Step Into	Step into the next method call at the currently executing line of code.
	Step Over	Step over the next method call (without entering it) at the currently executing line of code. The method will still be executed.
	Step Return	Return from a method which has been stepped into. The remainder of the code that was skipped by returning is still executed.
	Use Step Filters	Enables/disables the step filters functionality.

Note:

The Debug View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Debug**.

Variables View [PHP Debug Perspective]

The Variables view displays information about the variables associated with the stack frame selected in the Debug View. Selecting a variable will display details in the detail pane below the view. Expanding the list under a variable will display its fields.



Note:

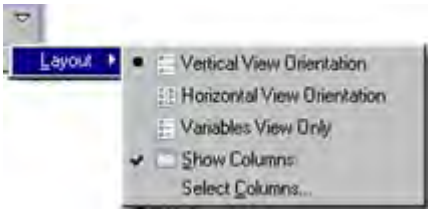
Right-click a variable and select **Watch** or **Create Watch Expression** to add the variable to the Expressions view.

Variables View Toolbar Commands

Icon	Name	Description
	Show Type Names	If selected, type names will be displayed.
	Show Logical Structure	Shows the logical structure.
	Collapse All	Collapses the list.

Variables View Menu Commands

The view's menu can be accessed through the view menu icon .



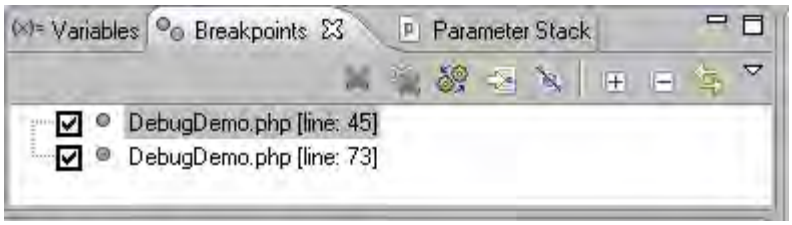
Name	Description
Layout	Defines the view's layout: ▪ Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view. ▪ Horizontal View Orientation - The details pane will be displayed to the right of the Variables view. ▪ Variables View Only - Only the Variables view will be displayed. ▪ Show columns - Divide the view into columns. ▪ Set Columns - Only available if "Show columns" is selected. Allows you to choose which of the following columns to display: • Name • Declared Type • Value • Actual Type

Note:

The Variables View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Variables**.

Breakpoints View [PHP Debug Perspective]

The Breakpoints view displays and allows you to monitor and control the breakpoints set in the files being debugged.

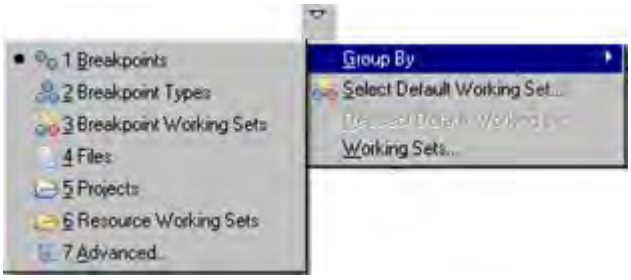


Breakpoints View Toolbar Commands

Icon	Name	Description
	Remove Selected Breakpoints	Removes the selected Breakpoints from the file.
	Remove All Breakpoints	Removes all Breakpoints from the file.
	Show Breakpoints Supported By Selected Targets	If selected, only breakpoints supported by the current 'debug target' will be displayed. For example, iIf a PHP file is being debugged, only PHP breakpoints will be displayed.
	Go to File for Breakpoint	Opens the resource in which the breakpoint is located.
	Skip All Breakpoints	If selected, all breakpoints will be skipped and execution will not stop.
	Expand All	Expands all items in the list.
	Collapse All	Collapses all items in the list.
	Link with Debug View	If selected, clicking a breakpoint will link with the Debug view.

Breakpoints View Menu Commands

The view's menu can be accessed through the view menu icon .



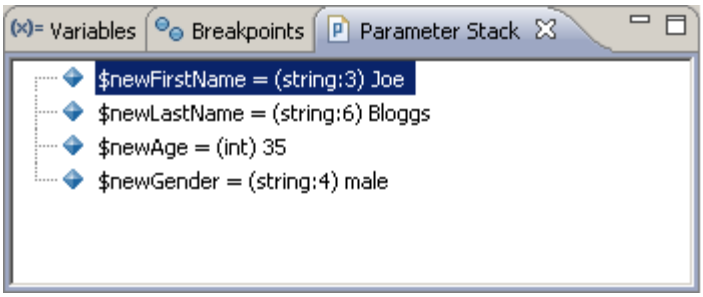
Name	Description
Group By	▪ Breakpoints ▪ Breakpoint Types ▪ Breakpoint Working Sets ▪ Files ▪ Projects ▪ Resource Working Sets ▪ Advanced
Select/Deselect Default Working Set	Allows you to choose the default breakpoint working set from the Default Working Set dialog.
Working Sets	Opens the Working Sets dialog.

Note:

The Breakpoints View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Debug | Breakpoints**.

Parameter Stack View [PHP Debug Perspective]

The Parameter Stack view displays the parameters executed when stepping into a function during the debugging process.



The following information can be gathered from the Parameter Stack view:

- Called Parameters - The called parameters as written in the line or code.
- The Main Calling Line of Code - The line number in which the calling statement occurred (in parentheses).
- Parameter Values - Shows the parameter values that were passed in the function call.

Note:

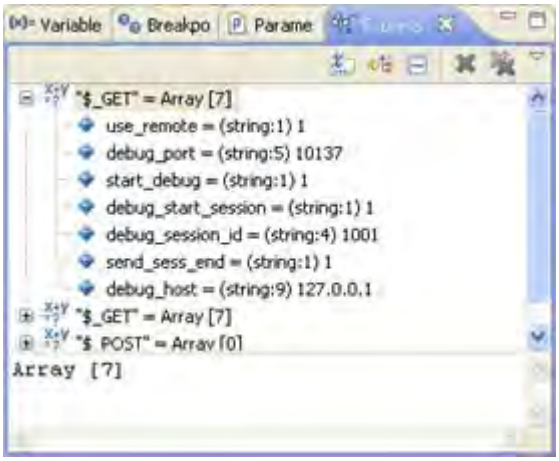
The Parameter Stack View [PHP Debug Perspective] is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | PHP Tools | Parameter Stack**.

Expressions View [PHP Debug Perspective]

The Expressions view will not open by default when a debugging session is launched, but only when you have selected to watch a variable or create a watch expression.
To watch a variable, right-click a variable in the editor or from the variables view and select **Watch** or **Add Watch Expression**. The Expressions view will open and the variable will be added to it. The variable's information will be updated as the debugging process continues.

Note:
To manually open the Expressions view, go to **Window | Show View | Debug | Expressions**.

The Expressions view allows you to monitor certain variables which you have decided to 'watch' during the debugging process. Selecting a variable will display details in the detail pane below the view. Expanding the list under a variable will display its fields.

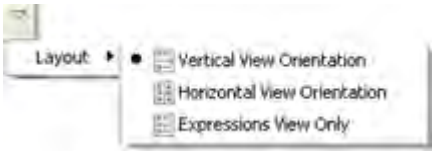


Expressions View Toolbar Commands

Icon	Name	Description
	Show Type Names	Displays type names
	Show Logical Structure	Displays the logical structure.
	Collapse All	Collapses the list.

Expressions View Menu Commands

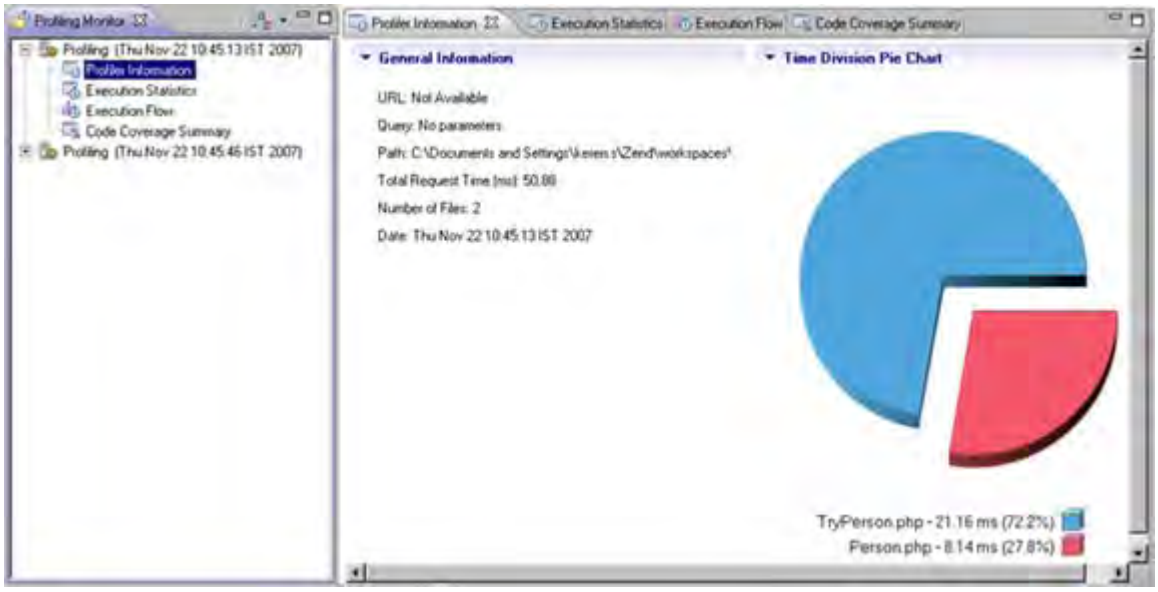
The view's menu can be accessed through the view menu icon .



Name	Description
Layout	Defines the view's layout: ▪ Vertical View Orientation - The details pane will be displayed at the bottom of the Variables view. ▪ Horizontal View Orientation - The details pane will be displayed to the right of the Variables view. ▪ Expressions View Only - Only the Watched Variables pane will be displayed.

PHP Profile Perspective

The PHP Profile Perspective can be launched automatically when a Profile session is run. It allows you to view all the information relevant to your scripts.

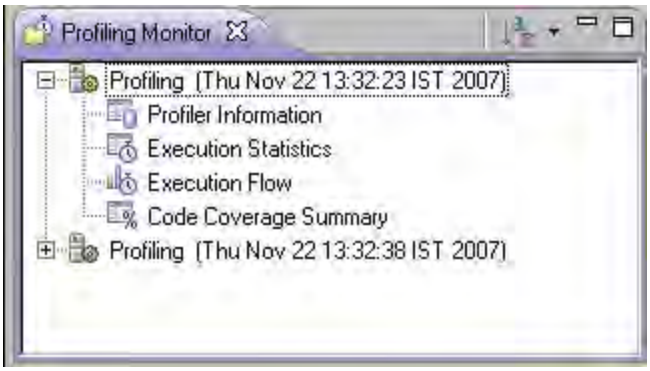


The PHP Profile Perspective contains the following views:

- Profiling Monitor
- Profiler Information
- Execution Statistics
- Execution Flow
- Code Coverage Summary

Profiling Monitor View

The Profiling Monitor view displays a list of previously run Profiling sessions.
Expanding the list under a Profiling session allows you to select a Profiling view to display.



Profiling Monitor View Toolbar Commands:

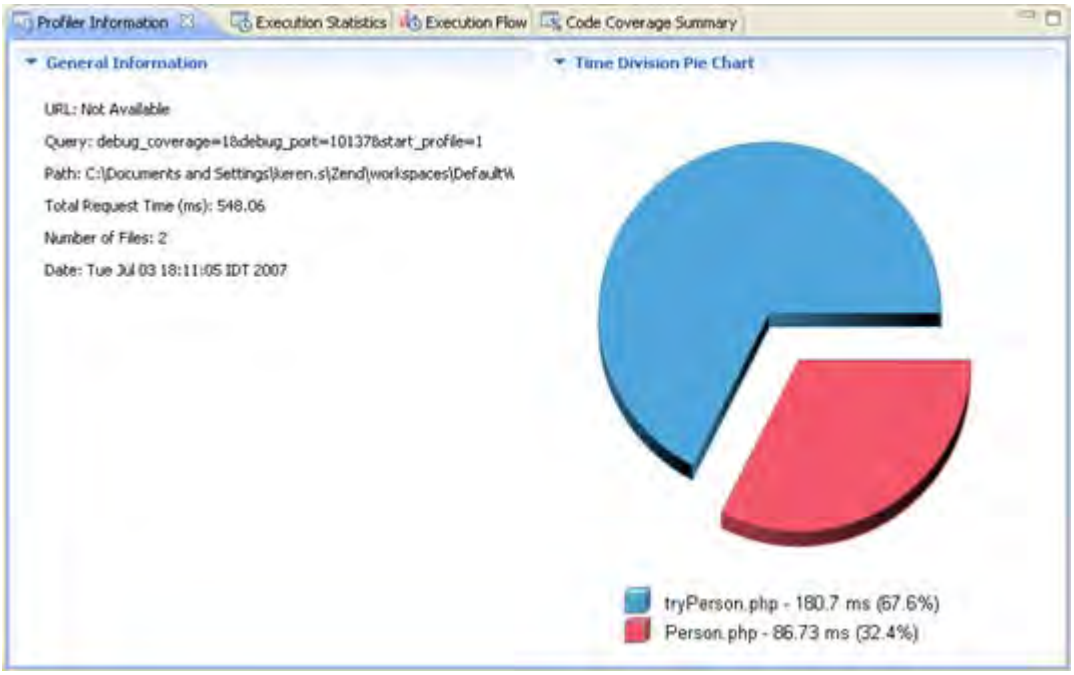
Icon	Name	Description
	Delete Session	Deletes a Profiling session from the list. This will be enabled if a profiling session is selected.
	Sort Profile Sessions	Click the arrow next to the Profile Session to sort the Profile Session list by date or time.

Note:

The Profiling Monitor view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Profiling Monitor.

Profiler Information View

The Profiler Information view provides general information about profiling sessions such as: duration, date, number of files constructing the requested URL and more. In addition, a Pie Chart is generated that shows the time division of the files in the URL.



The right side of the view displays time division in a pie chart and the left side provides the following information:

- URL - The URL analyzed (if applicable)
- Query - The specific query parameters
- Path - The location of the first file called
- Total Request Time - Total processing time for the entire page
- Number of Files - Number of files processed
- Date - Date and time that the profiling took place

Note:

The Profiler Information View view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Profiling Monitor.

Execution Statistics View

The Execution Statistics view displays the list of files that were called during the profiling process and detailed information on processing times for elements within the files.

Function	Calls Count	Average Own Time	Own Time(s)	Others Time(s)	Total time(s)
tryPerson.php					0.180701
main	1	0.180701	0.180701	0.086728	0.267429
Person.php					0.086728
Person					0.086722
__construct	3	0.028747	0.086240	0.000116	0.086356
getId	3	0.000004	0.000013	0.000000	0.000013
setFirstName	3	0.000016	0.000047	0.000000	0.000047
getFirstName	3	0.000003	0.000009	0.000000	0.000009
setLastName	3	0.000005	0.000016	0.000000	0.000016
getLastName	3	0.000003	0.000010	0.000000	0.000010
setAge	3	0.000006	0.000017	0.000000	0.000017
getAge	3	0.000004	0.000013	0.000000	0.000013

The window contains statistics relevant to each element as follows:

- Function - The name and location of the function.
- Calls Count - The number of times that the function was called.
- Average Own Time - The average duration without internal calls.
- Own Time(s) - The net process duration without internal calls.
- Others Time(s) - Time spent on calling other files.
- Total Time(s) - The total time taken to process.

Note:

Click the 'Show as percentage' button on the toolbar to see the statistics as percentages rather than times.

Right- clicking a function in the list gives you the option to 'Open Function Invocation statistics'. This will open a view with statistics about the selected function, the functions it was invoked by and functions that it invoked.

Execution Statistics View Toolbar Commands:

Icon	Name	Description
	Filters...	Click the arrow next to the icon to select to display only the results with: • Highest 10 own time • Highest 10 calls • Highest 10 total time • Highest 10 average time • -Or- No filter. Click the icon itself or select Manage Filters from the list to launch the Edit filter dialog which allows you to create or edit your own filter conditions.
	Expand/Collapse all	Expands/collapses the list.
	'Show as Percentage'	Toggles the view to show your times in seconds or percentages.
	Group by File	Sorts the list by file.
	Group by Class	Sorts the list by class.
	Group by Function	Sorts the list by function.

Note:

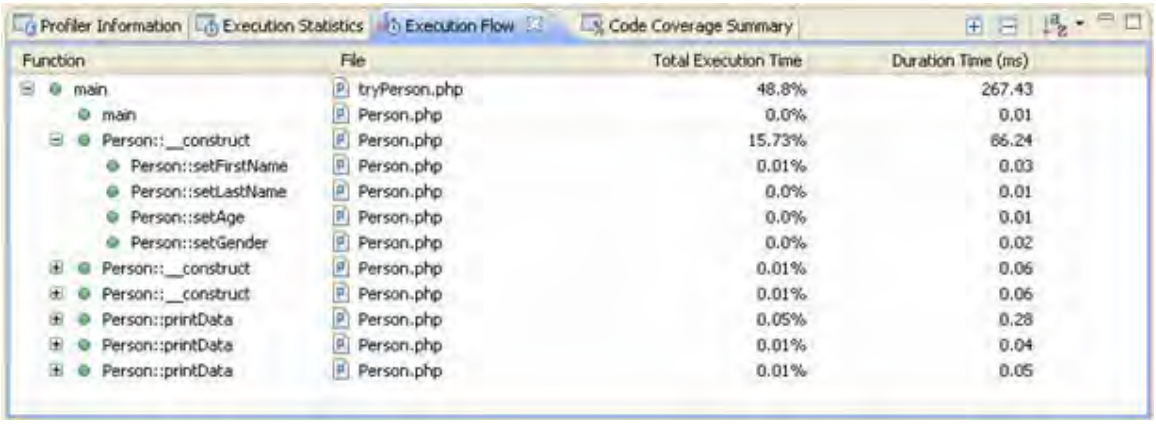
The Execution Statistics view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Execution Statistics.

Execution Flow View

The Execution Flow view shows the flow of the execution process and summarizes percentages and times spent on each function.

The view displays the following information for each function:

- Function - Function name
- File - The file in which the function is located
- Total Execution Time - Percent of time taken per function.
- Duration Time - Time taken per function. In milliseconds.



Right-clicking a function in the list gives you the option to:

- View Function Call - Opens the selected function call in the editor.
- View Function Declaration - Opens the selected function declaration in the editor.
- Open Function Invocation statistics - Opens a view with statistics about the selected function, the functions which the selected function was invoked by, and the functions invoked by the selected function.

Execution Flow view Toolbar Commands

Icon	Name	Description
	Expand/Collapse all	Expands/collapses the list.
	'Show as Percentage'	Toggles the view to show your times in seconds or percentages.
	Sort Profile Sessions	Click the arrow next to the Profile Session to sort the Profile Session list by the Order in which the functions were executed or by Duration Time.

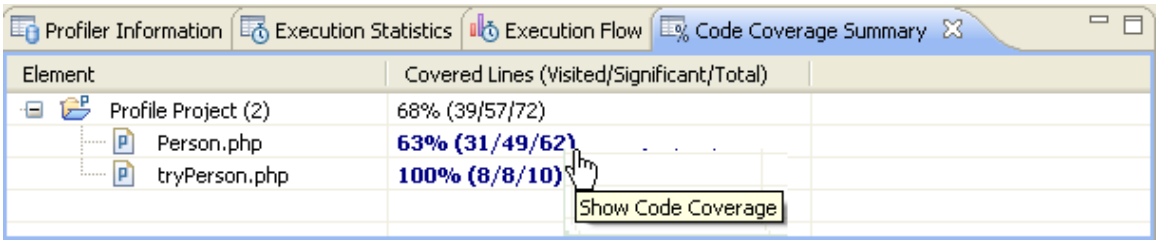
Note:

The Execution Flow view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Execution Flow.

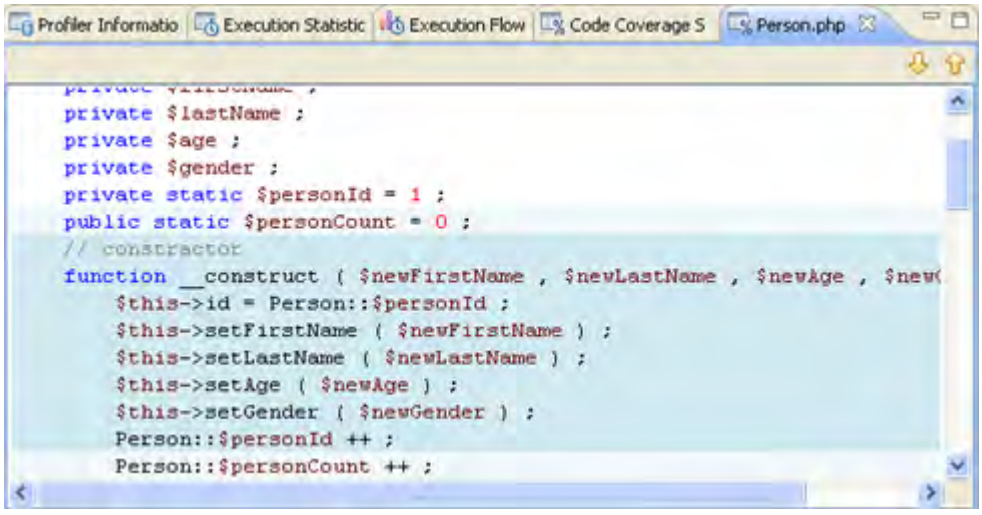
Code Coverage Summary View

The Code Coverage Summary view displays a summary of the lines of code that were covered during the Profiling process.
It contains the following columns:

- Element - The file / project that was called.
- Covered Lines (Visited / Significant / Total) - Percentage of lines covered within each file. (Visited = Number of lines covered / Significant = number of lines that were significant to the function call / Total = Total number of lines in the file.)



Clicking on the 'Covered lines' percentages will open an editor containing the file, with the covered lines highlighted:



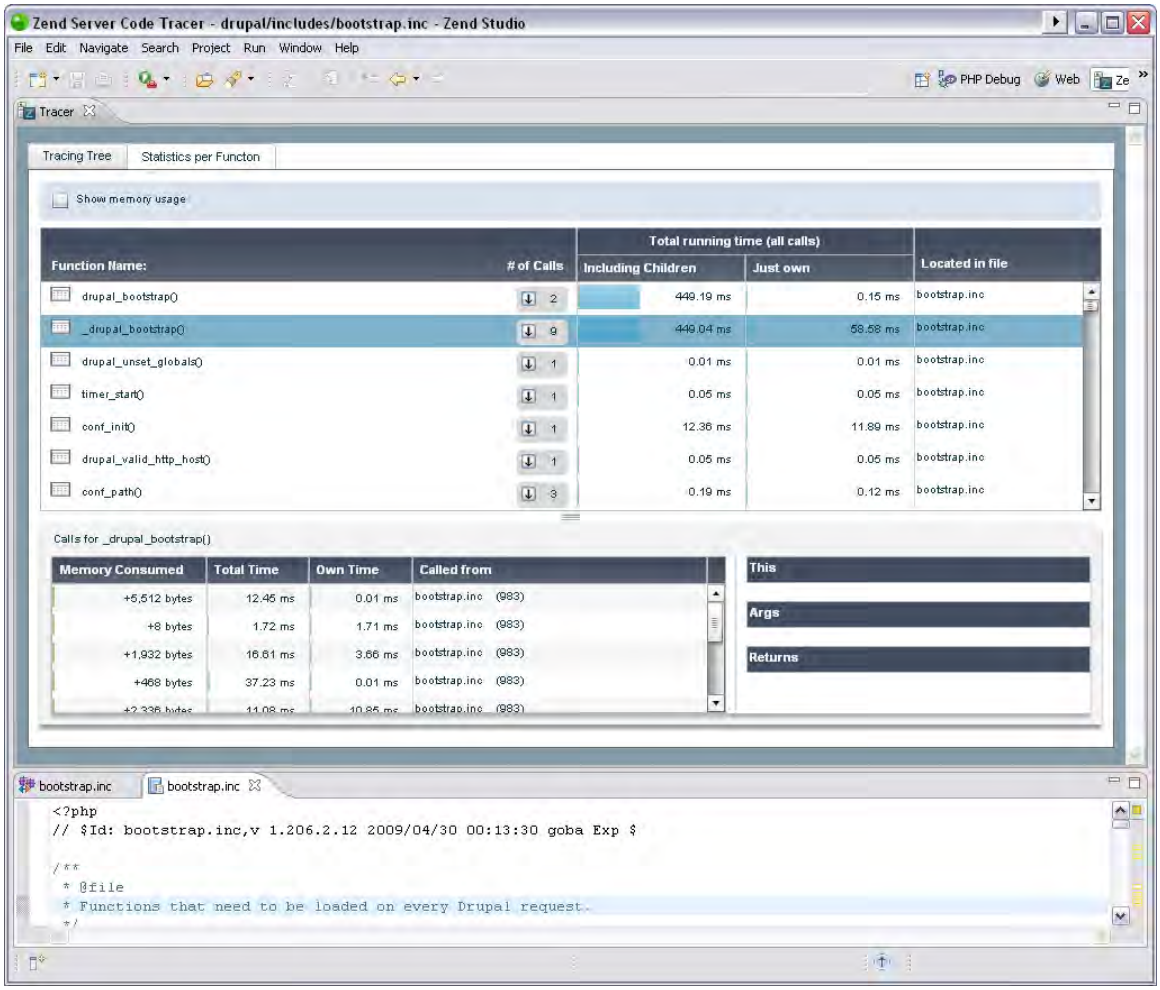
Note:

The Code Coverage Summary view is displayed by default as part of the Profiling Perspective. To manually open the view, go to Window | Show View | Other | PHP Profiler | Code Coverage Summary.

Zend Server Code Tracer Perspective

The Zend Server Code Tracer perspective allows you to use the Zend Server Code Tracing feature. Integrating Zend Server Code Tracing into Zend Studio allows you to open the source of the execution data inside of your environment. This feature is useful in resolving time performance issues, memory performance issues, and workflow errors.

The perspective is accessed by going to **Window | Open Perspective | Other | Zend Server Code Tracer**.

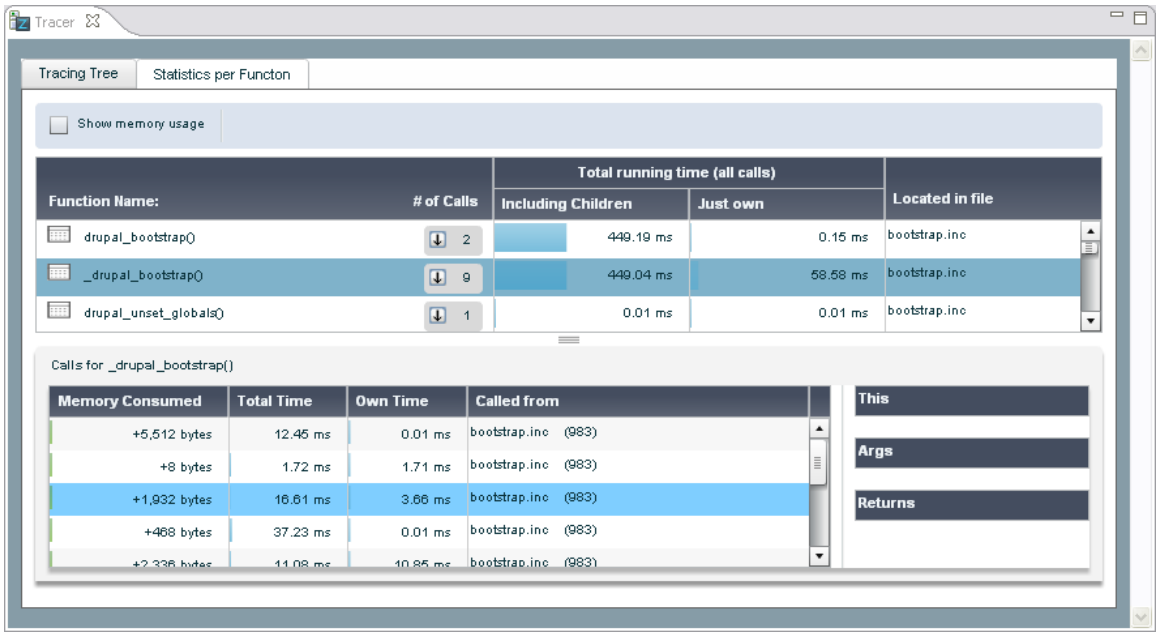


The Zend Server Code Tracer perspective contains the following views:

- Tracer View

Tracer View

The Tracer view displays the Imported Zend Server Event File.



The Tracer view includes the following tabs:

- Tracing Tree - Displays the call tree for a selected event or trace file. For more information see Code Tracing Tree in the Zend Server Online Documentation.
- Statistics per Function - A table based display that provides a statistical perspective of the data captured in the request. For more information see Code Tracing Statistics in the Zend Server Online Documentation.

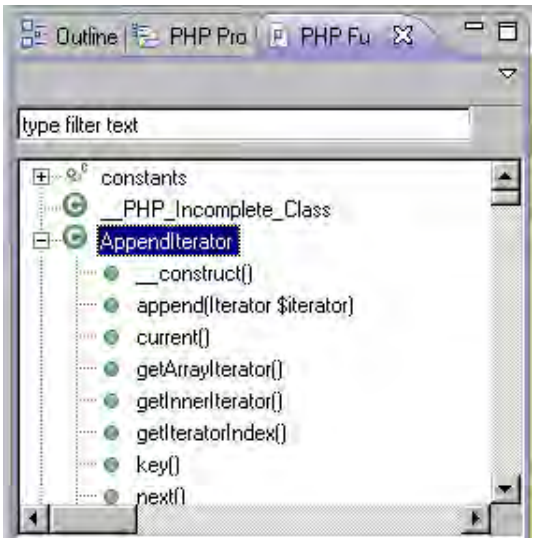
Note:

The Tracer View is displayed by default as part of the Debug Perspective. To manually open the view, go to **Window | Show View | Other | Zend Servers | Tracer**.

Additional Views

PHP Functions View

The PHP Functions view lists most commonly used PHP Classes, Constants and Iterators. The PHP Functions view can be used in order to easily add functions into your scripts. To add a function to your code, simply place the cursor in the required position in the Editor and double-click the required element from the list.



Right-clicking a function in PHP Functions view and selecting Open Manual will open an online version of the PHP manual with an explanation about most of the functions on the list.
A new browser window will open with an explanation of the function from the PHP Manual.

Note:

If the browser opens with a 'Cannot find server' error message, it means the function does not have a description assigned to it in the PHP Manual.

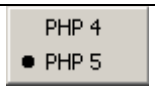
Sites for viewing the PHP Manual can be added and edited from the PHP Manual Preferences page.

PHP Functions View Toolbar commands

Icon	Name	Description
	Filter Text box	Allows you to find a particular function. Start typing the function name. Relevant results will be displayed below it.

PHP Functions View Menu Commands

The view's menu can be accessed through the view menu icon .

Icon	Name	Description
	PHP Version selection	Allows you to choose between PHP4 and PHP5 functions.

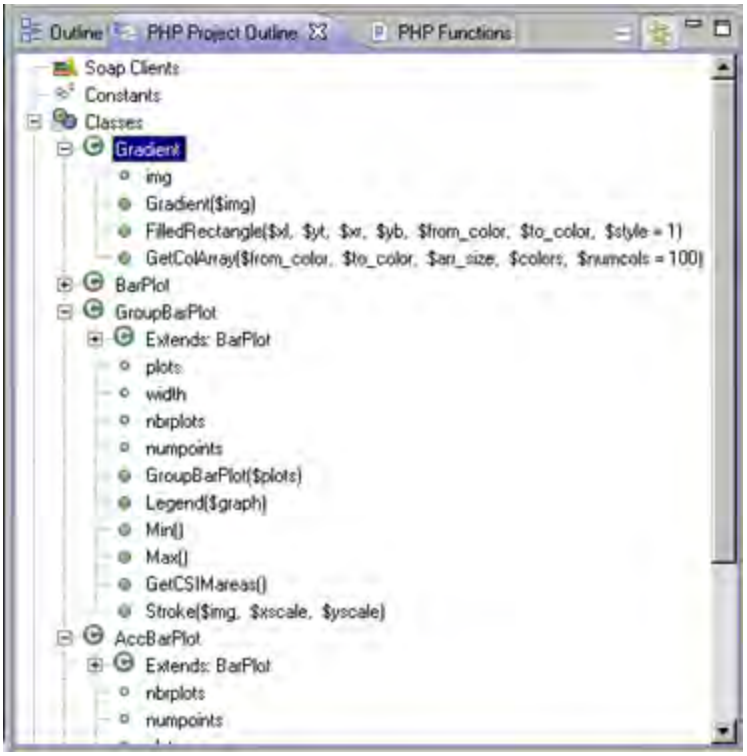
Note:

The PHP Functions View is displayed by default as part of the PHP Perspective. To manually open the view, go to Window | Show View | Other | PHP Tools | PHP Functions.

PHP Project Outline View

The PHP Project Outline view displays a list of Soap Clients, Namespaces (PHP 5.3 only), Constants, Classes and Functions for all files within the selected project.
Selecting an element in the PHP Project Outline view will open the relevant file in the editor.

To access the view, go to Window | Show View | Other | PHP Tools | PHP Project Outline.



PHP Project Outline View Toolbar commands

Icon	Name	Description
	Collapse All	Collapses the list of elements
	Link with Editor	If selected, elements will immediately be displayed in the editor when they are single-clicked. If unselected, elements will be displayed in the editor when they are double-clicked.

PHP Perspective Menus

Zend Studio's menu bars and toolbars offer a range of useful and easily accessible functionality.

Note:

The options available through the menu and toolbars will vary depending on which perspective is currently active.

To configure the menu options for the active perspective, go to Window menu and select Customize Perspective.

The PHP Perspective will by default display the following menu options:

Menu Option	Description
File	Carries out various functions on active files and folders, as well as organizing current files and creating new items. **The File menu options are:** New, Open File, Close, Close All, Save, Save As, Save All, Revert, Move, Rename, Refresh, Convert Line Delimeters To, Print, Switch Workspace, Import, Export, Properties, Last Viewed Files and Exit.
Edit	Normal text editing functionality, as well as features such as tasks and bookmarks which are specifically related to editing code. **The Edit menu options are:** Undo Text Change, Redo Text Change, Cut, Copy, Paste, Delete, Select All, Find/Replace, Find Next, Find Previous, Incremental Find Next, Incremental Find Previous, Add Bookmark, Add Task, Show Tooltip Description, Word Completion, Quick Fix and Set Encoding.
Source	Organizes your scripts by adding or removing comments and formatting the script to make it more easily viewable. **The Source menu options are:** Toggle Comment, Add Block Comment, Remove Block Comment, Format Document and Format Active Elements.
Refactor	Edits names and locations of files and resources while maintaining the links between the files. **The Refactor menu options are:** Organize Imports, Rename and Move.
Navigate	Navigates through your scripts in order to find relevant resources, information and text. **The Navigate menu options are:**

Menu Option	Description
	Go Into, Go To, Open Declaration, Open PHP Element, Open Resource, Show In, Next Annotation, Previous Annotation, Last Edit Location, Go to Line, Back and Forward.
Search	Searches for text or PHP elements in your workspace. **The Search menu options are:** Search, File and Text.
Project	Carries out different functions on your projects. **The Project menu options are:** Open Project, Close Project, Build All, Build Project, Build Working Set, Clean, Build Automatically, Generate PHPDoc.., Encode Project, and Properties.
Run	Gets maximum efficiency and accuracy from your files and projects through analyzing and testing your code using the Debugging, Profiling and Run functionality. **The Run menu options are:** Toggle Breakpoint, Toggle Line Breakpoint, Toggle Method Breakpoint, Toggle Watchpoint, Skip All Breakpoints, Remove All Breakpoints, Run History, Run As, Open Run Dialog, Debug History, Debug As, Open Debug Dialog, Run, Debug, Profile, Profile History, Profile As, Profile, External Tools, Debug URL and Profile URL.
Window	Customizes your workspace display. **The Window menu options are:** New Window, New Editor, Open Perspective, Show View, Customize Perspective, Save Perspective As, Reset Perspective, Close Perspective , Close All Perspectives, Navigation, Working Sets, Web Browser and Preferences.
Help	Gives access to the most updated information on all aspects of Zend Studio, as well as allowing access to software updates and registration locations so that you can get the most out of the product. **The Help menu options are:** Welcome, Help Contents, Search Dynamic Help, Key Assist, Tips and Tricks, Software Updates, Register, Tip of the Day and About Zend Studio.

File Menu

The File Menu allows you to carry out various functions on active files and folders, as well as organizing current files and creating new items.

The options available from the File menu are:

Name	Shortcut	Description
New	Alt+Shift+N	Creates various items and types of files. To see the list of new items that can be created through this menu, see the "New" subtopic. Note: The options available in the New Menu for the current perspective can be configured through the Customize Perspectives option in the Window Menu.
Open File		Opens a previously created file in an editor view. If the file is stored in your workspace's active projects, the file will be displayed in PHP Explorer / Navigator views.
Close	Ctrl+W	Closes the active file's editor. If there are unsaved changes in the file, you will be prompted to save the file before closing.
Close All	Ctrl+Shift+W	Closes all open editor windows. If there are unsaved changes in the file, you will be prompted to save them before closing.
Save	Ctrl+S	Saves changes made to the active file.
Save As		Allows you to specify the file name/location when saving the active file.
Save All	Ctrl+Shift+S	Saves all open files.
Revert		Reverts the state of the active file back to its last saved version.
Move		Moves the currently selected file to a different folder / project. Marking the 'Update References' check box in the Move dialog will apply the refactoring feature which will automatically updates all references to the file in other files. Click Preview to see all changes that will be made as a result of the move. Click OK to implement your changes and move the file.

Name	Shortcut	Description
		All references to the file will be automatically updated to reflect its new location. For more on moving files using the refactoring feature, see "Moving Files".
Rename	F2	Renames the currently selected file. Marking the 'Update References' check box in the Move dialog will apply the refactoring feature which will automatically update all references to the file in other files. Click Preview to see all changes that will be made as a result of the rename. Click OK to implement your changes. All references to the file will be automatically updated to reflect the change. For more on renaming files using the rename feature, see "Renaming a file".
Refresh	F5	Refreshes the Navigator views when external changes have been made.
Convert Line Delimeters To		Selects the preferred line ending style. Choices of Windows, Unix and Mac styles.
Print	Ctrl+P	Prints the active file.
Switch Workspace		Allows you to open an alternate workspace. Using this feature will restart Zend Studio with an alternate workspace displayed in Navigator / PHP Views. This is useful if you want to open files and projects situated in a different location or if you want to save files and projects to a different location.
Import		Imports various types of items into your workspace. For a list of the different types of items you can import, divided into categories, see "Import".
Export		Exports and creates different types of items from your workspace into various locations. For a list of the different items you can export from your

Name	Shortcut	Description
		workspace, divided into categories, see "Export".
Properties	Alt+Enter	Displays a screen with information on the active file, including it's path, type, location, size and when it was last modified. From here the text file encoding type can also be configured, and whether the file is read-only, archive or derived can be set.
Last viewed files		Lists the last viewed files for easy access.
Exit		Shuts down Zend Studio. If there are unsaved changes in the file, you will be prompted to save them before exiting.

New Submenu

The New submenu is available under File | New from the Menu Bar.

Note:
The options available in the New Menu for the current perspective can be configured through the Customize Perspectives option in the Window Menu.

The options available under the New submenu are:

Name	Description
PHP Project	Creates a new PHP project within your workspace, with PHP configuration settings allowing full PHP functionality.
Zend Framework Project	Creates a new Project with Zend Framework's libraries in the Include Path and files to create a basic "Hello, World!" application. For more on Zend Framework, visit the Zend Framework site at http://framework.zend.com For more on using Zend Framework with Zend Studio, see "Zend Framework Integration".
Project ...	Creates a new project in your workspace.
PHPUnit Test Case	Creates a new PHPUnit Test Case. See PHPUnit Testing for more information.
Task	Creates new repositories using the Task Repository view.
PHPUnit Test Suite	Creates a new PHPUnit Test Suite.. See PHPUnit Testing for more information.
PHP File	Creates a new file with PHP tags. Allows full PHP functionality.
Folder	Creates a new folder within a project. The new folder can be linked to a folder in the files system by clicking on the Advanced button in the new folder creation dialog. Using this option will insert an existing folder into your workspace folder. Any changes made to the files and folders in your workspace will automatically be reflected in the local versions of the files in your file system.
File	Creates a new file resource.
CSS	Inserts a cascading style sheet into a project.
HTML Page	Creates a new HTML file within a project, which allows the utilization of HTML functionality.

Name	Description
XML	Creates a new XML file within a project, which allows the utilization of XML functionality.
Zend Framework Item	Opens a New Zend Framework Item wizard which allows you to launch the Wizards for creating a Zend Controller, Zend Model, Zend Module, Zend View and Zend View Helper.
Class	Inserts a new PHP Class within existing or new files, including the required modifiers, Superclasses, interfaces, method stubs, comments etc.
Interface	Creates a new PHP Interface within existing or new files. The PHP Interface creation wizard allows you to include PHP Doc Blocks in your interface, as well as extending other interfaces.
Remote Folder	Creates a new Remote Folder with a remote system connection, See Viewing and Editing FTP Files in your Workspace for more information.
Java Script	Creates a new JavaScript file.
Example	Creates the following example projects in your workspace: ▪ **XML** - Inserts an XML example project into the workspace. ▪ **Zend Framework** - Zend Framework is a high quality open source framework for developing Web Applications and Web Services with PHP. The Zend Framework is a collection of common PHP classes and infrastructure which sits above the PHP layer. It packages classes and code, used for common functions such as connecting to databases and creating PDF's, into one easy-to use application. For more on using the Zend Framework example, see the Zend Framework tutorial. This is contained in a readme.html file within the readme folder of the ZendFrameworkExample project. To view the file, right-click it in PHP Explorer view and select Open With | Web Browser. For more information on Zend Framework, see http://framework.zend.com.

Name	Description
Other	Allows access to all other types of items not in the main list. To configure which items will be available from the main list, go to Window menu | Customize Perspective. The options in the list are divided into categories: ▪ **Class** ▪ **Folder** ▪ **Interface** ▪ **Java Project** ▪ **Java Project from Existing Ant Buildfile** ▪ **Plug-in Project** ▪ **General** - File, Folder, Project, Remote Folder, Unitled Text File ▪ **Connection Profiles** - Connection Profile, Connection Profile Repository ▪ **CVS** - CVS Repository Location, Projects from CVS ▪ **Eclipse Modeling Framework** - EMF Model, EMF Project, Empty EMF Project ▪ **Example EMF Model Creation Wizards** - Ecore Model, Ecore to Ecore Model, Ecore to XML Model, XSD Model, ZSD to Ecore Model ▪ **Java** - Annotation, Class, Enum, Interface, Java Project, Java Project from Existing Ant Buildlife, Package, Source Folder, DBUnite Test Case, Scrapbook Page, JUnit Test Case, JUnite Test Suite, Servlet Test Case ▪ **Java** - DbUnit - DbUnit Test Case ▪ **Java Emitter Templates** - Convert Projects to JET Projects ▪ **JavaServer Faces** - JSF Library ▪ **JPA** - JPA Project ▪ **PHP** - PHP Class, PHP File, PHP Interface, PHP Project, Untitled PHP Document, Zend Controller, Zend Framework Project, Zend Model, Zend Module, Zend View ▪ **PHP Unit - PHPUnit Test Case, PHPUnit Test Suite** ▪ **Remote System Explorer - Connection** ▪ **SQL Development** - SQL File ▪ **SVN** - Projects from SVN, Repository Location ▪ **Web - CSS, HTML, JavaScript, Static Web Project**

Name	Description
	▪ **Web Services - WSDL** ▪ **XML -** DTD, XML, XML Schema ▪ **Examples** - GEF (Graphical Editing Framework) Plug-ins, XML, Zend Framework

Import Submenu

The Import submenu is available under File | Import from the Menu Bar.

The options available under the Import submenu are:

Name	Description
General	**Archive File** - Extracts files from the archive file into the workbench. **Existing Projects into Workspace - Projects from the local file system.** **File System - Files from the local file system. Browse to the folder in which the file is sitting and click OK. A list of files within that folder will be displayed to allow you to choose the required ones.** **Preferences** - Import preferences from a preferences file on the local file system into the workbench.
CVS	**Projects from CVS** - Imports projects by connecting to a CVS repository. Once a project has been imported from CVS, changes can be made to projects and files which can then be committed to update the CVS repository. A CVS repository connection needs to be configured before using this function. For more information, see Using CVS.
PHP Profiler	**Profile Session** - Imports a profile session file into the workspace.
Remote Systems	**Remote file system- Imports resources from a remote file system.**
Run/Debug	**Breakpoints - Imports a brakpoint working set.** **Launch Configurations - Imports a debug launch configurations file.**
SVN	**Projects from SVN - Imports projects by connecting to an SVN repository. Once a project has been imported from SVN, changes can be made to projects and files which can then be committed to update the SVN repository.** An SVN repository connection needs to be configured before using this function. For more information, see "Working with SVN".

Name	Description
Team	**Team Project Set - Imports a description of the repository and version control information for a set of projects.** For more on Team Project Sets, see'Sharing your workspace setup using Project Sets 'Sharing your workspace setup using Project Sets'.
XML	**XML Catalog - Imports an XML Catalog file.**
Zend Imports	**Import from Zend Studio 5.x - Imports projects from Zend Studio 5.x** Use this function if you have projects in your Zend Studio 5.x which you would like to open and edit in Zend Studio 7. Do not keep Zend Studio 5.x open while importing Zend Studio files. See Migrating Projects From Zend Studio 5.X for more information.
Zend Server	**Server Preferences - Imports server preferences from a server preferences (.zsc) file.** **Zend Server Event File - Imports a Zend Server Event File for debugging. See Importing Events Into Zend Studio for more information.**

Export Submenu

The Export submenu is available under File | Export from the Menu Bar.

The options available under the Export submenu are:

Name	Description
General	**Archive File** - Exports files from the Workbench to an archive file in the local file system. **File System - Export files from your workspace to your local file system.** **Preferences - Export preferences from the Workbench. In the Export Preferences wizard, select the preferences to export and the location of the preferences (.epf) file to which you want to export them.**
PHP	**PHP Doc - Creates a PHPDoc from your projects or files.** See PHPDocs for more information.
PHP Profiler	**HTML Report - Create an HTML report from a profile session. To use this feature, a profile session must first be run on a file by going to Run menu | Profile As and selecting the relevant profiling configuration. For more on profiling, see the Profiling topic** **Profile Session** - Creates an xml document from your profile session. To use this feature, a profile session must first be run on a file by going to Run menu | Profile As and selecting the relevant profiling configuration. For more on profiling, see the Profiling topic.
Remote Systems	**Remote file system - Exports resources to a remote file system.**
Run/Debug	**Breakpoints - Exports breakpoints from the workbench to a breakpoint file. A list of available breakpoints will be displayed in the Export Breakpoint wizard. Select the relevant breakpoints and the name and location of the file to which they should be exported.** Note: The Breakpoint (.bkpt) file will not appear in Navigator / PHP Explorer views.

Name	Description
	Launch Configurations - Imports a debug launch configurations file.
Tasks	**Task List and Contexts - Export all Task List data, useful for migrating workspaces.**
Team	**Team Project Set - Exports Imports a description of the repository and version control information for a set of projects.** For more on Team Project Sets, see ''Sharing your workspace setup using Project Sets'.
XML	**XML Catalog - Exports an XML Catalog file.**
Zend Servers	**Server Preferences - Imports server preferences from a server preferences (.zsc) file.**

Edit Menu

The Edit menu contains normal text editing functionality, as well as features such as tasks and bookmarks which are specifically related to editing code.

The options available from the Edit menu are:

Name	Shortcut	Description
Undo Text Change	Ctrl+Z	Undoes the last text edit in the active file
Redo Text Change	Ctrl+Y	Redoes the last text edit in the active file.
Cut	Ctrl+X	Cuts the selected section of text.
Copy	Ctrl+C	Copies the selected section of text to the clipboard.
Paste	Ctrl+V	Pastes text from the clipboard.
Delete	Delete	Deletes the selected section of text.
Select All	Ctrl+A	Selects all text within a file.
Find / Replace	Ctrl+F	Finds and replaces text within the active file.
Find Next	Ctrl+K	Goes to the next instance of an item selected in the editor.
Find Previous	Ctrl+Shift+K	Goes to the next instance of an item selected in the editor.
Incremental Find Next / Previous	Ctrl+J Ctrl+Shift+J	Finds character strings after / before the cursor within the active file. To use this feature, press Ctrl+J and the first few letters of the required string. The relevant text will be highlighted in the file. While in this mode, the up and down cursor keys can be used to navigate between matches. The search can be cancelled by pressing left, right, Enter or Escape.
Add Bookmark...		Inserts a bookmark into your script. Bookmarks are used to easily navigate to specific sections in your scripts. You can attach a descriptive name to each Bookmark which can be later seen in a tooltip next to the Bookmark.

Name	Shortcut	Description
		Bookmarks are indicated by a bookmark icon in the left margin. Open the Bookmark view (Window | Show View | Other |General | Bookmarks) to navigate between existing Bookmarks.
Add Task...		Inserts a task into your script. Tasks are used as reminders to the programmer. For maximum effectiveness, tasks should be placed next to the section of code on which the action will be implemented. Open the tasks view (Window | Show View | Tasks) to navigate between existing tasks.
Show Tooltip Description		Shows the value of a hover that would appear at the current cursor location. The dialog shown is scrollable and does not shorten descriptions.
Word Completion	Alt+/	Completes a word being typed. Enter the first few letters of the word and press Alt+/ to complete the word. Completes a prefix to a word occurring in all currently open editors or buffers.
Quick Fix	Ctrl+1	Displays possible quick fix options for problems in the Problems view. To use this option, first select a problem in the Problems view. Note: This option will not always be available.
Set Encoding		Changes the file encoding used to read and write the file in the active editor.

Source Menu

The Source menu allows you to organize your scripts by adding or removing comments and formatting the script to make it more easily viewable.

The options available from the Source menu are:

Name	Shortcut	Description
Override/Implement Methods		Launches the Override/implement Method Wizard to override/implement methods defined in the selected class's parent or interface. See Overriding / Implementing Methods for more information.
Generate Getters and Setters		Launches the Generate Getters and Setters Wizard to automatically create getter and setter functions for variables within the selected class. See Generating Getters and Setters for more information.
Toggle Comment	Ctrl+/	Comments or uncomments a line by adding or removing "//" characters. Comments are used for adding text to your script to explain sections of code. Commented text will not be run as part of your code. To use this feature, select the line and press Ctrl+/. See Commenting Code for more information.
Add Block Comment	Ctrl+Shift+/	Comments a block by adding "/*" and "*/" characters to either side of the code. To use this feature, select the block and press Ctrl+Shift+/. See Commenting Code for more information.
Remove Block Comment	Ctrl+Shift+\	Removes a block comment. To use this feature, place the cursor anywhere within the comment and click Ctrl+Shift+\. See Commenting Code for more information.
Format Document	Ctrl+Shift+F	Auto formats a script to organize it into an easily readable format. To format your code, place your cursor anywhere within the editor view and press Ctrl+Shift+F.

Name	Shortcut	Description
		Appropriate line breaks and indents will be added. You can configure your auto-formatting options through the Formatter Preferences page, accessible from Window | Preferences | PHP | Formatter. See Formatting Code for more information.
Format Active Elements	Ctrl+I	Only formats selected code. To format active elements, select the required code to format and press Ctrl+I. Appropriate line breaks and indents will be added to the active elements. You can configure your auto-formatting options through the Formatter Preferences page, accessible from Window | Preferences | PHP | Formatter. See Formatting Code for more information.

Refactor Menu

The Refactor menu allows you to edit names and locations of files and resources.

For more on refactoring, see "Using Refactoring".

The options available from the Refactor menu are:

Name	Shortcut	Description
Move	Alt+Shift+V	Moves a file to a different folder. To move a file, select it from the PHP Explorer view. A Move dialog will open. Select the required folder and click Preview to see all changes that will be made as a result of the Move. Click OK to implement your changes and move the file. All references to the file will be automatically updated to reflect its new location.
Rename	Alt+Shift+R	Renames a file or element. To rename a file, select it from the PHP Explorer view. To rename an element within a file select it from the PHP Explorer view or highlight it in the editor view. A rename dialog will open. Enter the new name and click Preview to see all changes that will be made as a result of the rename. Click OK to implement your changes. All references to the file / element will be automatically updated to reflect the change.
Extract Method	Alt+Shift+M	Extracts methods from selected code.
Extract Local Variable	Alt+Shift+L	Extracts variables from selected code. See Extracting Variables for more information.

Note:

Refactoring options will only be available from within PHP Explorer view and not from Navigator view.

Using the Navigator view's move/rename functions will not update any referenced instances of the file/element.

Navigate Menu

The Navigate menu allows you to navigate through your scripts in order to find information and text.

The options available from the Navigate menu are:

Name	Shortcut	Description
Go Into		Goes into a selected folder so that only that folder's contents will be displayed in the Navigator view. Note: This will not work in PHP Explorer view.
Go To		**Back** Displays the previously displayed hierarchy in the Navigator view.
		Forward **Returns to the display from which the back button was pressed in the Navigator view.**
		Up one level Will go up one level in the hierarchy in the Navigator view.
		Resource Goes to a resource within the files and folders displayed in the Navigator view. Enter the first few letters of the a resource in the Go To Resource dialog, and select the required one from the list. Note: This functionality will not work in the PHP Explorer view.
	Ctrl+Shift+P	**Matching Bracket** Jumps to a bracket's pair. Clicking to the right of a bracket will highlight its matching pair. To jump to the matching bracket, press Ctrl+Shift+P.
Open Declaration	F3	Goes to the declaration of an item selected in the editor.
Open Method		Opens a Method in the workspace. See Opening Types/Methods for more information.
Open Type		Opens a Type in the workspace. See Opening Types/Methods for more information.
Open Type in Hierarchy		Displays a selected Type in a hierarchy. See Viewing Type Hierarchies for more information.

Name	Shortcut	Description
Open Resource...	Ctrl+Shift+R	Opens files within the same project as the active file. An Open Resource dialog will appear. Enter the first few letters of the required file to see a list of matching files. Select the required file and click OK to open it in an editor window.
Show In	Alt+Shift+W	**PHP Explorer -** Displays the current active file in the PHP Explorer view. **Navigator** - Displays the current active file in the Navigator view. **Outline -** Displays an element in the Outline view.
Quick Type Hierachy		Displays a selected Type in a Quick Type Hierarchy. See Viewing Type Hierarchies for more information.
Next / Previous Annotation	Ctrl+. Ctrl+,	Goes to the next / previous annotation in the script. Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings. Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included. Possible annotation types are Bookmarks, Diff Additions, Diff Changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings.
Show Zend Server Event List		Opens the Zend Server Event List. This is only applicable if a Zend Server has been configured in the PHP Servers Preferences page. See Zend Server Integration for full information on the Zend Server.
Last Edit Location	Ctrl+Q	Jumps to the last location that was edited.
Go to Line...	Ctrl+L	Allows you to go to a specific line in the active editor.
Back / Forward	Alt+Left Alt+Right	Scrolls through last viewed sections in active and previously edited files in the current session.

Search Menu

Allows you to Search for text or PHP elements in your workspace.

The options available from the Search menu are:

Name	Shortcut	Description
Search	Ctrl+H	Opens the PHP Search dialog. PHP Search enables you to locate declarations of PHP Classes, Functions and Constants. For more, see "Searching for PHP Elements".
File		Opens the File Search dialog. File Search enables you to locate text in all files in your workspace. To run a File Search or Replace: 1. Enter the required text to be searched for. 2. Select the types of files to search in. Click 'Choose' for a full list of file type extensions. 3. Select whether to search in: • Workspace - the entire workspace • Selected resources - Select these in PHP Explorer view before opening the Source dialog • Enclosing projects - The projects which the selected resources are in. • Working Set - Click 'Choose' to select the required Working Set. 4. To search for the string, click Search. Search results will be displayed in the Search view. To replace the string, click Replace. The Replace dialog will open. **Note:** By default, the PHP Search dialog will be tabbed with the File Search dialog. To make the PHP Search dialog unavailable, click Customize within the File Search dialog and unmark the PHP Search dialog option.
Text	Ctrl+Alt+G	**Workspace** - Searches the Workspace for text selected in the editor.

Name	Shortcut	Description
		Project - Searches the current active project for text selected in the editor. **File -** Searches the current active file for text selected in the editor. **Working Set -** Searches the current active Working Set for text selected in the editor.

Project Menu

The Project menu allows you to carry out different functions on your projects, including open, close, build and encode.

Note:

For more information on the build process, see the topic in the Workbench User Guide. This can be accessed from inside Zend Studio for Eclipse's Online Help (by going to Help | Help Contents), or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

The options available from the Project menu are:

Name	Shortcut	Description
Open Project		Opens the currently selected project. This option is enabled when a closed project is selected.
Close Project		Closes the currently selected project. Closing a project does not cause it to be deleted from the file system. A closed project will still be displayed in PHP Explorer view with a closed project icon , but its resources are no longer accessible from within the Workbench. Closing projects takes up less memory and speeds up the build process.
Build All	Ctrl+B	This command manually invokes an incremental build on all projects in the Workbench. This is only available if automatic build is not selected (see below).
Build Project		This command manually invokes an incremental build on any resources in the currently selected project that have been affected since the last build. This is only available if automatic build is not selected (see below).
Build Working Set		This command manually invokes an incremental build on any resources in a working set that have been modified since the last build. This is only available if automatic build is not selected (see below).
Clean...		Invokes a clean build. This will discard all previous build

Name	Shortcut	Description
		results.
Build Automatically		Performs an incremental build whenever resources are saved. Selecting this option will disable all other manual build options. **Note:** The build function can be configured by selecting General | Workspace from the preferences dialog (Windows | Preferences).
Zend Tool	Ctrl+2	Opens the Zend Tool Floating WIndow for executing Zend_Tool commands. See Using the Zend Tool Console more information.
Generate PHPDoc		Creates a PHPDoc from your projects or files. A PHPDoc is an online document, organized in a book format, allowing easy viewing of all elements within your code.
Encode Project		Encodes the selected project using Zend Guard. This option will be enabled once Zend Guard is installed and configured in Zend Studio. Zend Guard integration can be configured through the Zend Guard preferences dialog, accessed from Window | Preferences | PHP | Zend Guard.
Properties		Opens the project's properties dialog which allows you to view and configure various settings for the project, including resource information, Builders, Code Analyzer Properties, Formatter, Includes Mapping, PHP Debug, PHP Include Path, PHP Interpreter, PHP Java Bridge, PHP Task Tags, Project References, Run/Debug Settings, Task Tags and Validation. See PHP Project Properties for more information on all the properties available for a PHP project.

Run Menu

The Run menu allows you to get maximum efficiency and accuracy from your files and projects through analyzing and testing your code using the Debugging, Profiling and Run functions.

Running a file or application will display the output in the Browser and Debug Output views, as well as displaying any error or warning messages in the console view.

Debugging a file or application allows you to view the output and any error notices, as well as information about various elements, at various stages while the file is run. For more information on Debugging, see Debugging Files and Applications.

Profiling a file or application allows you to detect bottlenecks in scripts by locating problematic sections of code.
For more information on Profiling, see Profiling Files and Applications.

The options available from the Run menu are:

Name	Shortcut	Description
Toggle Breakpoint	Ctrl+Shift+B	Adds / removes breakpoints from your script. Breakpoints are used to stop the debugging process at certain key places throughout your code.
Toggle Line Breakpoint		Adds / removes line breakpoints from your script.
Toggle Method Breakpoint		Adds / removes method breakpoints from your script. Method breakpoints are used to add conditions to breakpoints.
Toggle Watchpoint		Adds / removes a field watchpoint for the current selected variable in the Expressions View.
Skip All Breakpoints		Temporarily removes all breakpoints from your script so that the debugging process will not stop at them. Select this option again to return all breakpoints to the script.
Remove All Breakpoints		Removes all breakpoints from the current active file.
Run History		Displays and allows access to a list of previously launched Run configurations.
Run As		Lets you choose from running: on the server, as a PHP

Name	Shortcut	Description
		Script, or as a PHP WebPage.
Run Configurations...		Launches the Run dialog to create / edit Run configurations.
Debug History		Displays a list of Debug configurations so that they can be used for debugging.
Debug As		Lets you choose from debugging: on the server, as a PHP Script, or as a PHP WebPage. For more on these options, see "Debugging".
Debug Configurations...		Launches the Debug dialog to create /edit debugging configurations.
Run	Ctrl+F11	Launches the last Run configuration run.
Debug	F11	Launches the last Debug session run.
Profile		Launches the last Profile session run.
Profile History		Displays a list of Profile configurations so that they can be used for profiling.
Profile As		Lets you choose from profiling: on the server, as a PHP Script, or as a PHP WebPage. For more on these options, see "Profiling".
Profile Configurations...		Launches the Profile dialog to create / edit Profiling configurations.
External Tools		**Run As** If applicable, allows you to run the file using External tools. **Open External Tools Dialog** Opens the configuration dialog for running a file using external tools. **Organize Favorites** Opens a dialog allowing you to organize your external tools. See the External ToolsExternal Tools topic in the Workbench User Guide for more information.
Debug URL		Launches a Debug session for a specified URL.
Profile URL		Launches a Profile session for a specified URL.

Navigation Submenu

The Navigation submenu is available under Window | Navigation from the Menu Bar.

The options available under the Navigation submenu are:

Name	Shortcut	Description
Show System Menu	**Alt + -**	Shows the system menu for the active view. This contains options on how the view is displayed (Fast View, Detached, Restore, Move, Size, Minimize, Maximize, Close.)
Show View Menu	Ctrl+F10	Shows the active view's menus, containing functionality for each view.
Quick Access	Ctrl+3	Gives you quick access to a range of Eclipse options.
Maximize / Minimize Active View or Editor	Ctrl+M	Toggles between full screen and minimal views.
Activate Editor	F12	Switches to the editor view.
Next / Previous Editor	Ctrl+F6 Ctrl+Shift+F6	Switches to the next/previous open editor. Hold down the Ctrl key and press F6 to scroll between the editors. This command is similar to the Alt+Tab functionality in Windows.
Switch to Editor	Ctrl+Shift+E	Opens a dialog displaying all the open editors. Allows you to choose which editors to open, close or save.
Next / Previous View	Ctrl+F7 / Ctrl+Shift+F7	Switches to the next open view. Hold down the Ctrl key and press F7 / Shift + 7 to scroll between the views. This command is similar to the Alt+Tab functionality on Windows.
Next Perspective / Previous Perspective	Ctrl+F8 / Ctrl+Shift+F8	Switches to the last / next open Perspective.

Window Submenu

The Window menu allows you to customize your workbench display.

The options available from the Window menu are:

Name	Shortcut	Description
New Window		Opens the workspace in a new window. Clicking on the window's X icon will close only that window and not the whole workspace.
New Editor		Opens the active editor in a new window.
Open Perspective		Opens a selected perspective, containing a selection of views. See the Workbench User Guide for more on Perspectives. **Note:** Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).
Show View		Displays a selected view. See the Workbench User Guide for more on Views. **Note:** Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).
Customize Perspective		Configures settings for the active perspective, including settings for: - The quickly accessed settings on the New, Open Perspective and Show View submenus. - Which options appear in the menu and toolbar.
Save Perspective As..		After configuring the perspective, you can select to save it under a different name for future use.
Reset Perspective		Resets the perspective to its default view, menu bar and toolbar settings.
Close Perspective		Closes the active perspective and reverts back to the last viewed perspective.

Name	Shortcut	Description
Close All Perspectives		Closes all perspectives. No views or functionality will be available. Click the open perspective icon to open a perspective.
Navigation		Allows quick access and Navigation between views and perspectives. For a list of commands available from this menu option, see the "Navigation" subtopic.
Working Sets		Opens the working sets dialog, allowing you to edit or create Working Sets See PHP Working Sets for more information.
Preferences		Opens the preferences dialog to configure all aspects of the workspace. See the PHP Preferences section for on PHP Preferences.

Help Menu

The Help menu allows access to the most updated information on all aspects of Zend Studio, as well as allowing access to software updates and registration locations so that you can get the most out of the product.

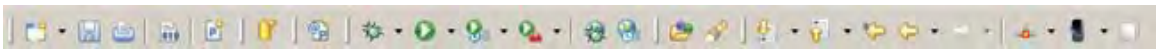
The options available from the Help menu are:

Name	Shortcut	Description
Welcome		Opens the welcome page containing the latest news from Zend as well as access to a range of tutorials for using Zend Studio functionality.
Help Contents		Opens the Online Help page.
Search		Opens a Search view for searching the online Help manual.
Dynamic Help		Opens Help Topics relevant to the current action in the workspace.
Key Assist...	Ctrl+Shift+L	Opens a list of all shortcut keys for Zend Studio functionality.
Tips and Tricks...		Opens the Tips and Tricks Help page for the Eclipse Platform and Eclipse Plug-In Development Environment.
Cheat Sheets...		Allows you to open a cheat sheet with quick explanations on how to carry out a variety of functions.
Register		Allows you to enter your Zend Studio license key to enable Zend Studio functionality. For more on purchasing a Zend Studio license, see the Zend Studio site at http://www.zend.com/en/products/studio. **Note**: System i users can register for a free i5 Edition of Zend Studio. See i5 Edition Extras for more information.
Zend Studio Support Center		Redirects you to the Zend Online Support Center.
Tip of the Day		Opens the Tip of the Day window.
Check for Updates		Searches for available software updates.
Install New Software...		Installs available software updates.

Name	Shortcut	Description
Get a License		Connects you to the Zend Studio site at http://www.zend.com/en/products/studio to learn more about purchasing a license and enabling Zend Studio functionality.
Protect your PHP Code!		Directs you to the Zend Guard site at http://www.zend.com/en/products/guard. Zend Guard protects your applications from reverse engineering and unauthorized customization by providing encoding and obfuscation. See Zend Guard Integration for more information.
Speed up your PHP Site!		Directs you to the Zend Server site at http://www.zend.com/en/products/server.Zend Server is the only PHP Web application server that supports the enterprise reliability and comprehensive performance features organizations need for business-critical applications. See Zend Server Integration for more information.
About Zend Studio		Opens the About dialog, displaying information about the current version of Zend Studio.

PHP Perspective Main Toolbar

The PHP Perspective's Main Toolbar offers shortcuts to frequently used functionality:



Shortcut Icon	Shortcut Keys	Name	Description
		New	Opens the New Wizard dialog. Clicking the arrow next to the icon lets you select to create a new Zend Framework Project, PHP Project, Project, Zend Module, Zend Controller, Zend Model, Zend View, PHPUnit Test Case, PHPUnit Test Suite, PHP Class, PHP Interface, Remote Folder, PHP File, Folder, CSS, HTML, XML, Example Project or Other resource.
	Ctrl+S	**Save**	Saves the active file.
	Ctrl+P	**Print**	Prints the active file.
	Ctrl+Alt+N	**New Untitled PHP Document**	Creates a new untitled PHP Document.
		Create New SQL Connection	Opens the New SQL Connection dialog.
		Generate PHPDoc	Opens the PHPDoc Generation wizard.
		Zend Server Event List	Displays the Event List for the configured Zend Servers. Clicking the arrow next to the icon allows you to select the required Zend Server. See Zend Server Integration for more information.

Shortcut Icon	Shortcut Keys	Name	Description
		Debug	Clicking the Debug Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options: ▪ Debug a previously executed launch configuration. ▪ Debug As... - Debug the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. ▪ Debug Configurations - Opens the Debug dialog. ▪ Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list.
		Run	Clicking the Run Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options: ▪ Run a previously executed launch configuration. ▪ Run As... - Run the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. ▪ Run Configurations - Opens the Run dialog. ▪ Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list.

Shortcut Icon	Shortcut Keys	Name	Description
		Profile	Clicking the Profile Button executes the last run configuration. Clicking the arrow next to the icon gives access to the following options: ▪ Profile a previously executed launch configuration. ▪ Profile As... - Profiles the active file as a PHP Script, PHP Web Page, or, when applicable, a PHPUnit Test. ▪ Profile Configurations - Opens the Debug dialog. ▪ Organize Favorites - Allows you to select which launch configurations should be added to your Favorites list. Your Favorite launches will be listed first in the launch configuration list.
		External Tools	Clicking the External Tools Button opens the External Tools Configuration dialog. Clicking the arrow next to the icon gives access to the following options: ▪ Run As - If applicable, allows you to run the file using External tools. ▪ External Tools Configurations - Opens the configuration dialog for running a file using external tools. ▪ Organize Favorites - Opens a dialog allowing you to organize your external tools. See the External Tools topic in the Workbench User Guide for information on configuring your Builder. **Note:** Additional user guides can be accessed from

Shortcut Icon	Shortcut Keys	Name	Description
			inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).
		Debug URL	Launches the Debug URL dialog.
		Profile URL	Launches the Profile URL dialog.
	Ctrl+H	**Search**	Launches the Search dialog.
	Ctrl+. Ctrl+,	**Next/Previous Annotation**	Navigates to the next / previous annotation in the script. Possible annotations are: Bookmarks, Diff additions, Diff changes, Errors, Info, Search Results, Spelling Errors, Tasks and Warnings. Click the arrow next to the next / previous annotation icon on the toolbar to configure which types of annotations should be included.
	Ctrl+Q	**Last Edit Location**	Jumps to the last location that was edited.
	Alt+Left Alt+Rigt	**Back/forward to last edited file**	Scrolls through the previous/next edited locations in all edited file in the current session.

Shortcut Icon	Shortcut Keys	Name	Description
		Enable Tunneling	Creates a Tunneling connection. Click the arrow next to the Tunneling icon to select the server for which you want to enable a Tunneling connection.
		Encode Project with Zend Guard	Encodes your project with Zend Guard. This will only be available when Zend Guard integration is enabled through the Zend Guard Preferences page.

PHP Preferences

Below is a list of the different PHP preferences which can be configured.

To access this menu, go to Window menu and select Preferences | PHP.

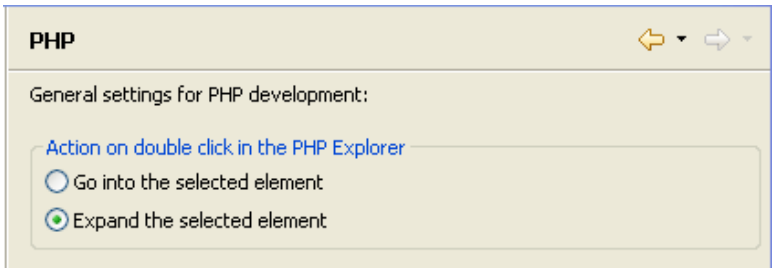
Preference	Description
PHP	Configures the display in PHP Explorer view.
Appearance	Configure the display of elements in Outline views.
Code Coverage	Preview code in the Code Coverage view with the current color and font settings.
Code Gallery	Define and add code galleries.
Code Refactor	Configure code re-factoring preferences.
Code Style	Expand the list to access the Code Templates and Formatter Preferences.
Code Templates	Configure generated code and comments.
Formatter	Set preferences for the auto-formatter.
Debug	Configure your debug preferences.
Installed Debuggers	Configure the settings for your installed debuggers.
Step Filtering	Configure the Step Filtering options when debugging.
Workbench Options	Configure the workspace's behavior when a debug session is launched.
Editor	Configure Smart Caret positioning.
Content Assist	Configure Code Assist preferences.
Code Folding	Configure the elements which will be folded by default.
Hovers	Configure the settings and shortcuts for the hover functionality.
Mark Occurrences	Configured the Mark Occurences settings.
Save Actions	Configures the behavior of whitespaces after saving.
Syntax Coloring	Set the font color for different elements.
Task Tags	Add and edit tasks tags.
Templates	Define and create templates for content assist.
Typing	Configure which items should be automatically completed.
Zend Framework	Configure display settings for the Zend Tool Floating Window and Console.

Preference	Description
New Project Layout	Configure the default project layout for new PHP projects.
PHP Executables	Add, remove or edit PHP executables definitions.
PHP Interpreter	Select the PHP version.
PHP Libraries	Add, remove, or edit PHP user libraries.
PHP Manual	Add PHP Manual sites.
PHP Servers	Add and edit PHP servers.
PHPUnit	Configure PHPUnit's Library path and port.
Semantic Analysis	Configure the severity for error and warning messages in different cases.
Zend Guard	Set Zend Guard's location.

PHP Preferences Page

The PHP Preferences page allows you to configure the hierarchy display in PHP Explorer view and set double-click behavior.

The PHP Preferences page is accessed from Window | Preferences | PHP.

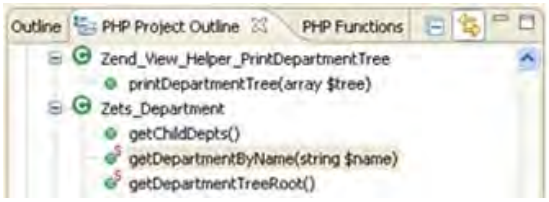

PHP
General settings for PHP development:
Action on double click in the PHP Explorer
Go into the selected element
Expand the selected element

To configure PHP Preferences:

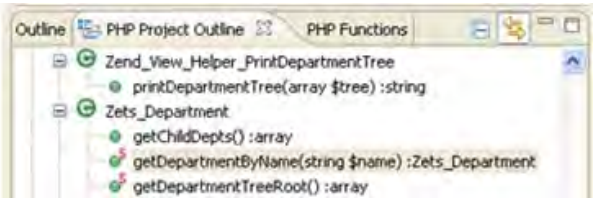
1. Select the required option:
   - Go into the selected element - PHP Explorer view will display only a folder's contents once it is double-clicked.
   - Expand the selected element - PHP Explorer view will expand a folder once it is double-clicked, leaving the other projects and folders visible in a tree diagram.
2. Click Apply to apply your settings.

Appearance Preferences

The Appearance preferences page allows you to select whether to show PHP Elements' method return types in the Outline and PHP Project Outline views. These will be displayed in brackets next to the element.

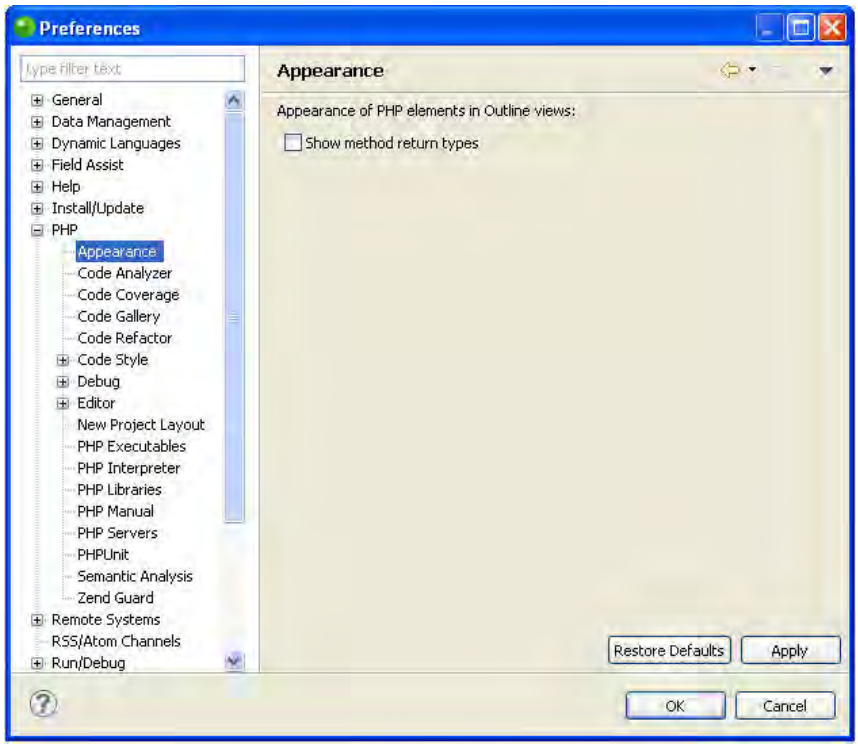


PHP Project Outline view without the method return types display



PHP Project Outline view with the method return types display

The Appearance Preferences page is accessed from Window | Preferences | PHP | Appearance.



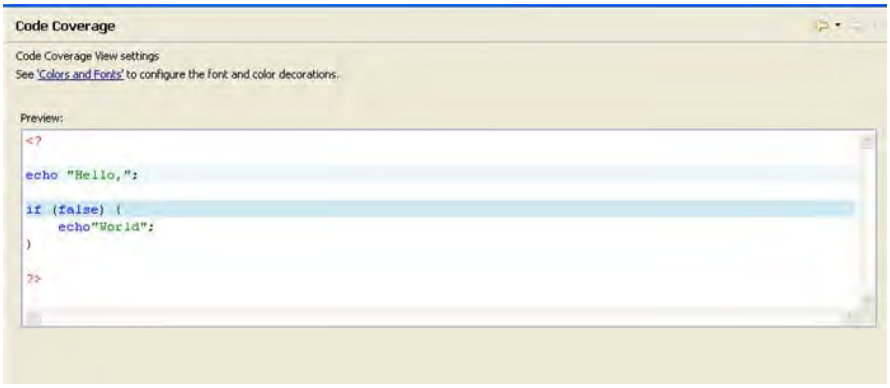
To display element's return type methods:

1. Mark the 'Show methods return types checkbox'.
2. Click Apply to apply your settings.

Code Coverage Preferences

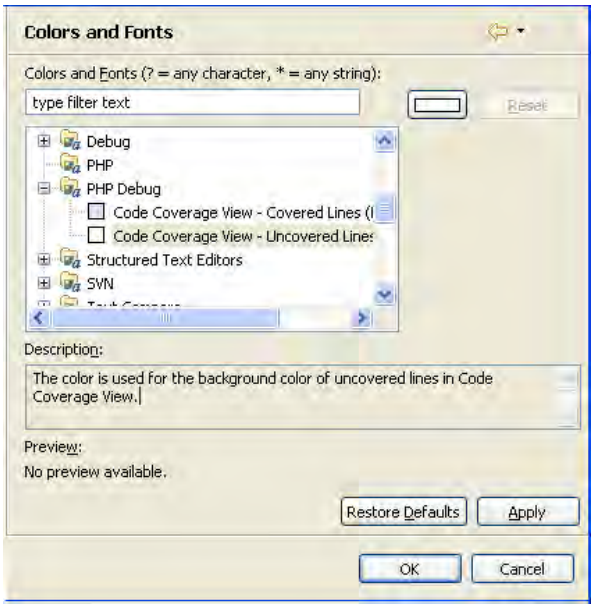
The Code Coverage Preferences page displays a preview of code in Code Coverage view, with the current color and font settings. Code coverage views are displayed when using various features such as profiling, unit testing and debugging to show which lines of code have been covered by this functionality.

The Code Coverage Preferences page is accessed from Window | Preferences | PHP | Code Coverage.



To configure code coverage colors and fonts:

1. Click the 'Colors and Fonts' link.
   The Colors and Fonts preferences page will be displayed, with the PHP Debug category open.



2. Select the required background color for covered lines and uncovered lines by selecting the relevant option and clicking the required color in the color selection box (top-right corner).
3. Click Apply.
4. The changes will be displayed in the Code Coverage preview page.

More color and font options can be configured by opening the preferences page (Window | Preferences) and selecting:

- General | Appearance | Colors and Fonts
- General | Editors | Text Editors | Annotation
- General | Editors | Text Editors | Quick Diff
- Run / Debug
- Run / Debug | Console
- Team | CVS | Console

Code Gallery Preferences

The Code Gallery preferences page allows you to add and edit code galleries.

Code Galleries are pre-defined code snippets sites. Code snippets can be used to easily insert pre-defined sections of code into your script.

The Code Gallery Preferences page is accessed from Window | Preferences | PHP | Code Gallery.

Code snippets can be accessed from the Code Gallery view (Window | Show View | Other | PHP Tools | Code Gallery).

To add a code gallery site:

1. Click Add.
2. Enter the URL of the required code gallery and click OK.

The new site will be added to the list and will be available from the Code Gallery view.

To edit a code gallery site:

1. Select the required gallery from the list.
2. Click Edit.
3. Change the required information and click OK.

To remove a code gallery site:

1. Select the required gallery from the list.
2. Click Remove.
3. Click OK to apply your settings.

The gallery will be removed from the list and will not longer be accessible from the Code Gallery view.

Note:

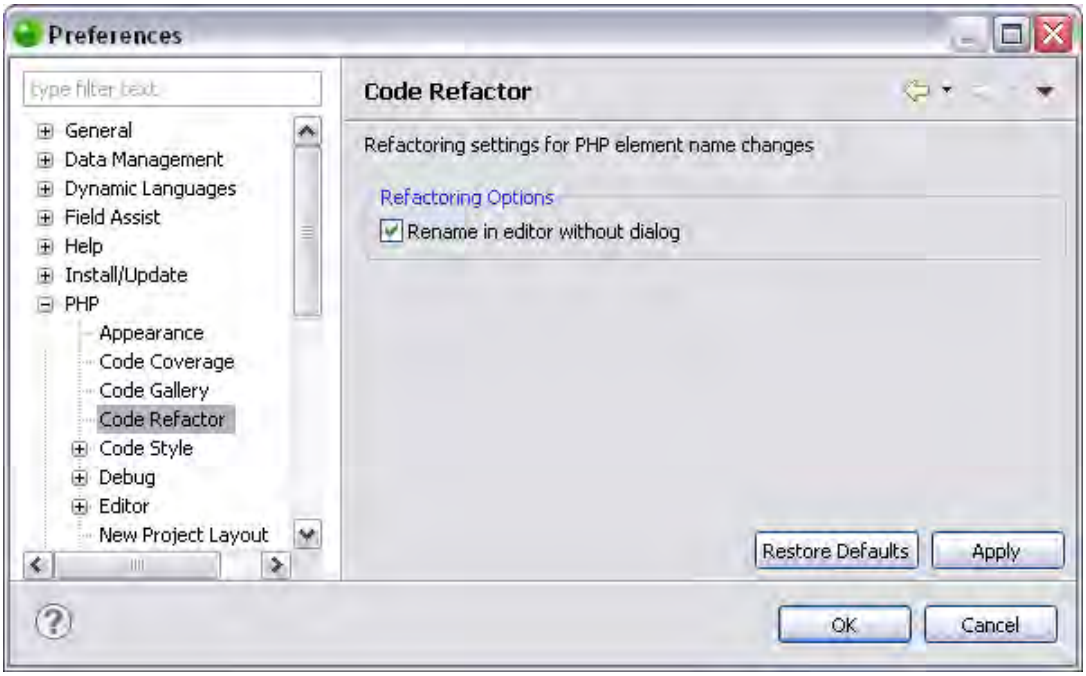
The User Code Gallery cannot be edited or removed.

Code Refactor Preferences

The Code Refactor Preferences page allows you to enable the in-place refactoring option. This option allows you to rename elements using the refactoring feature from within the editor rather than through the refactor dialog.

See the Refactoring topic for more information on refactoring or Renaming Elements

The Code Refactor Preferences page is accessed from Window | Preferences | PHP | Code Refactor.



To configure your Code Refactor settings:

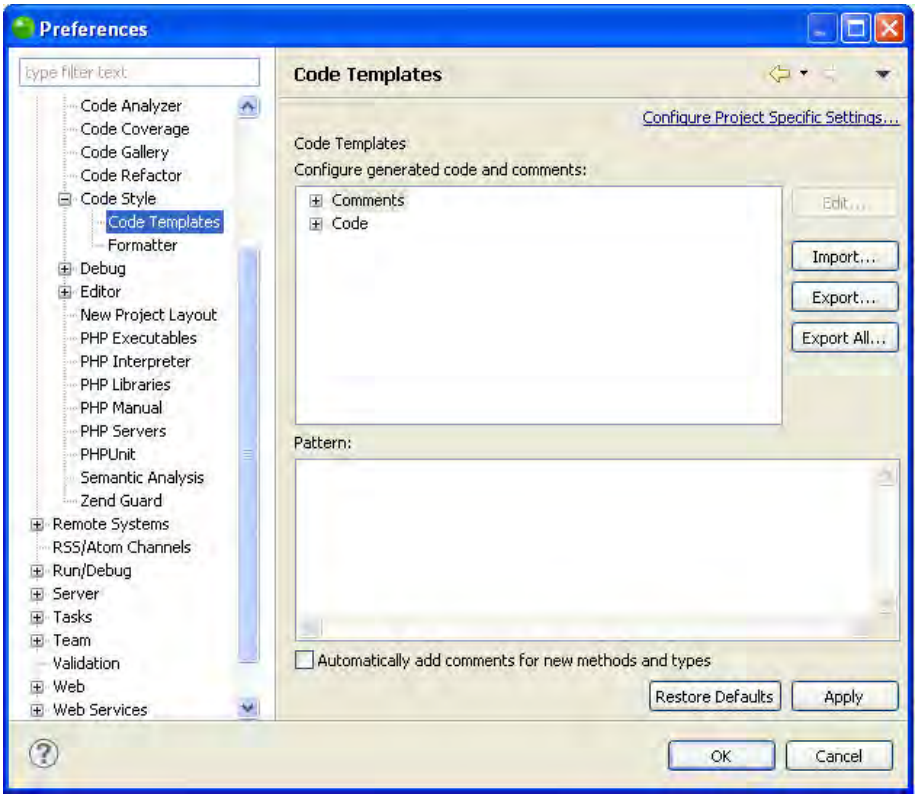
1. Mark the 'Rename in editor without dialog' checkbox to enable you to rename elements in the editor rather than through the refactor dialog.
2. Click Apply to apply your changes.

Code Style Preferences

Code Templates Preferences

The Code Templates Preferences page allows you to configure the code and comments that are automatically created for different types of elements.

The Code Templates Preferences page is accessed from **Window | Preferences | PHP | Code Style | Code Templates**.



To edit the pattern for a comment or code element:

1. Expand the list and select the required element.
2. Click **Edit**.
   The "Edit Template" dialog is displayed.
3. Edit the pattern as required and click **OK**.
4. If required, mark the 'Automatically add comments for new methods and types' box for comments to be automatically generated when new methods and types are created.

5. Click **Apply** to apply your changes.

Exporting and Importing Code Templates

Zend Studio enables you to export and import code templates, which are created within XML files in the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<templates>
<template
autoinsert="true" context="php" deleted="false"
description="description" enabled="true"
id="org.eclipse.php.ui.editor.templates.php.codetemplates.xxxcomment"
name="name">
/** * ${tags} */
</template>
</templates>
```

To import a code template:

1. Click Import to open the "Import Templates" browser.
2. Select the relevant XML file containing the template information.
3. Click **Open**.

The templates contained in the template.xml file will be imported into the list of Templates.

To export a code template:

1. Select the template(s) for export from the Template list.
2. Click **Export** to open the "Export Templates" dialog.
3. Select the location to save the XML file to.
4. Click **Save**.

An XML file will be created with the template information.

To export all code templates:

1. Click **Export All**.
2. Select the location to save the XML file to.
3. Click **Save**.

An XML file will be created with the template information.

Note:

If you selected more than one template to export, all of them will be present in the exported XML file. Each of the original Templates is bounded by: < template > </template>

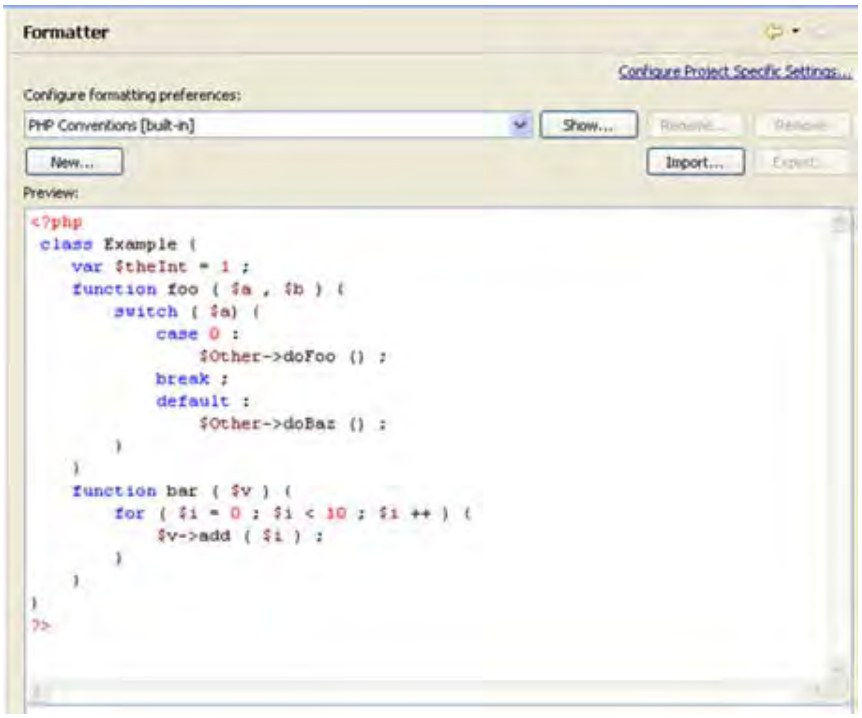
Formatter Preferences

Formatter Preferences

Zend Studio can auto-format scripts to organize them into an easily readable format.

The Formatter Preferences page allows you to customize the way it is formatted.

The Formatter Preferences page is accessed from Window | Preferences | PHP | Formatter.



The default formatting settings are based on the PHP Conventions settings.

Click Show to see these settings.

To create your own set of configuration settings:

1. Click New.
2. Enter a name for you profile.
3. Select which profile to base your new profile on. (This will duplicate these settings and allow you to edit them).
4. Ensure that the 'Open the edit dialog now' checkbox is marked and click OK.
5. The Edit dialog will open with the following tabbed option screens:
   - Indentation
   - Braces

- White Space
- Blank Lines
- New Lines
- Control Statements
- Line Wrapping

These are described in more detail below.

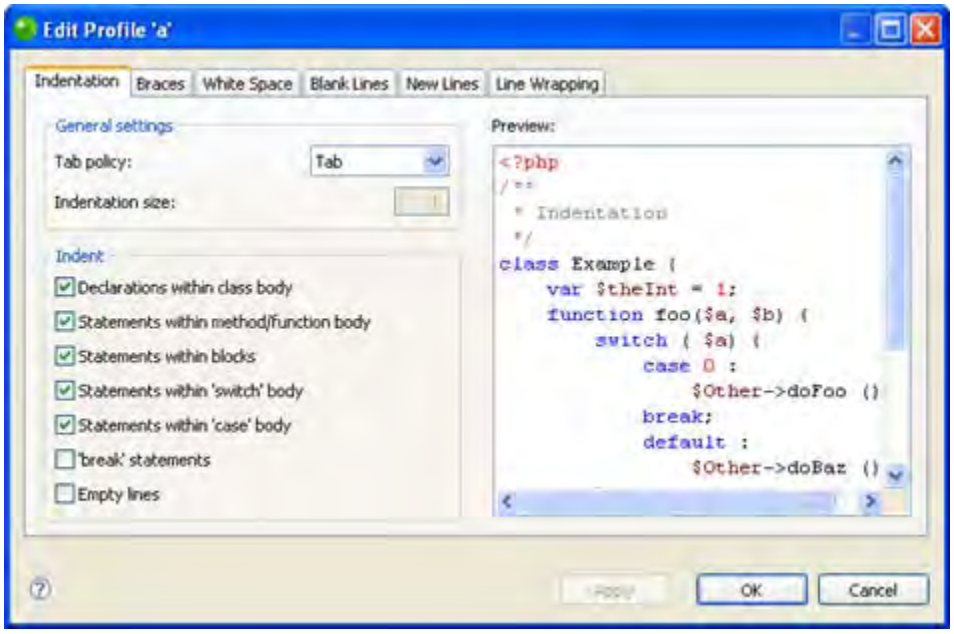
6. Once you have set the required options, click OK.

The new configuration will be added to the list.

The options available in the various formatter tabs are:

Indentation

The Indentation tab allows you to configure the indentation size and select which elements should be indented.



The Indentation tab contains the following options:

General Settings

- Tab Policy - Select Tab or Spaces from the drop-down list to set the indentation size for a tab.
- Indentation size - If you select spaces, enter the number of spaces to be created.

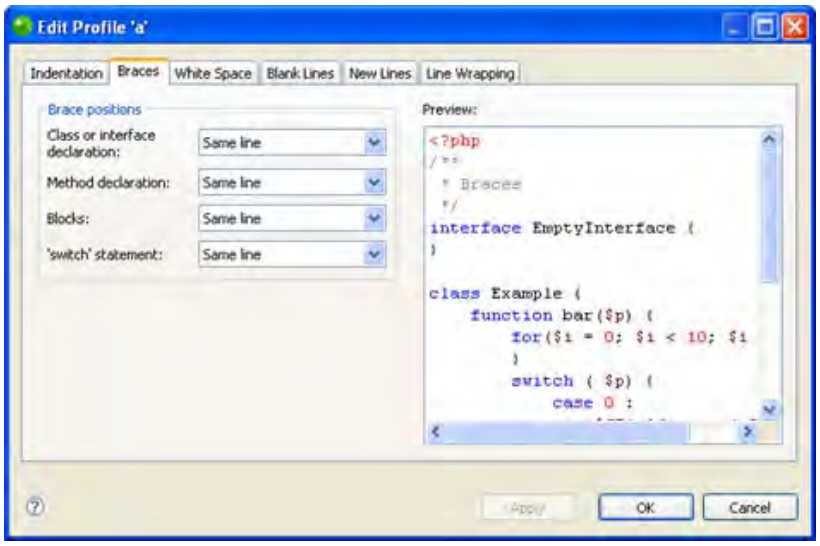
Indent

Select the elements to be indented by marking the relevant checkbox from the following options:

- Declarations within class body
- Statements within method/function body
- Statements within blocks
- Statements within 'switch' body
- Statements within 'case' body
- 'break statements
- Empty lines

Braces

The Braces tab allows you to configure brace positions.

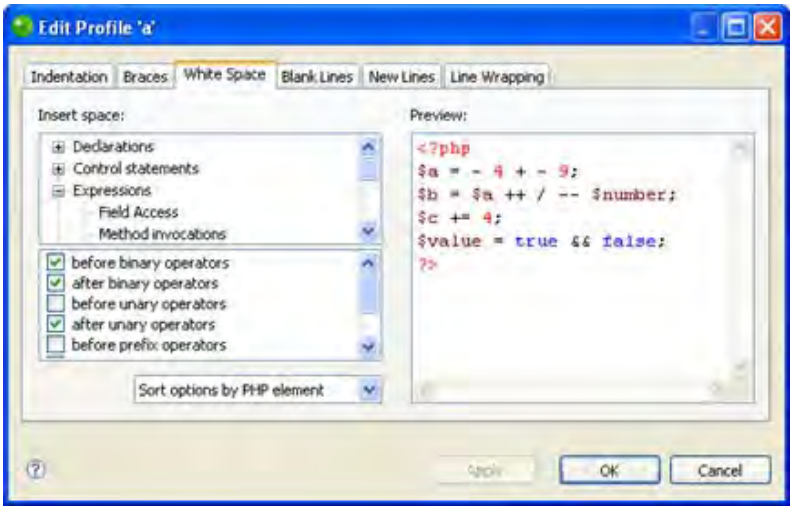


Choose the brace positions (Same line, Next line or Next line indented) for the following:

- Class or interface declaration
- Method declaration
- Blocks
- 'switch' statement

White Space

The White Space tab allows you to configure where spaces should be entered for declarations, control statements, expressions and arrays.



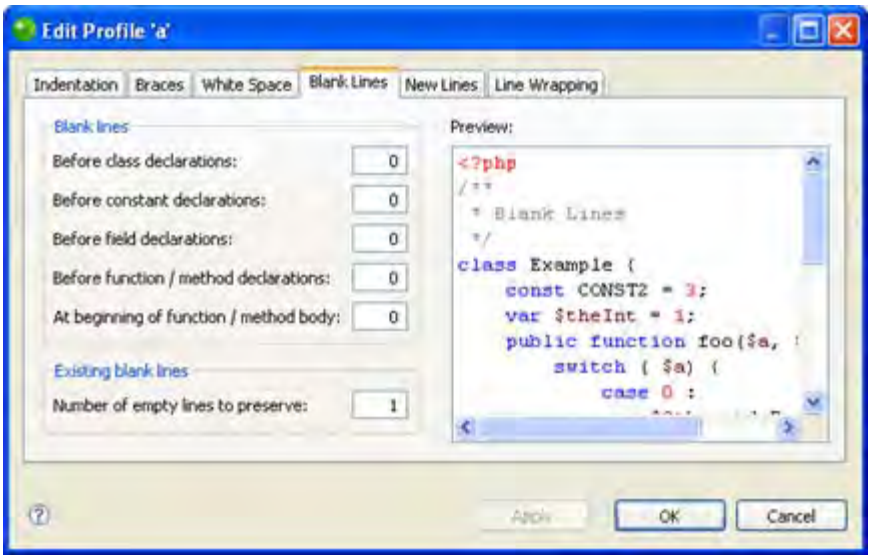
Expand each category by clicking on the + sign to configure which items are applied to each instruction.

Standing on an item will display a list of possible syntax conditions in which white spaces will be inserted. Select the required conditions by marking the relevant checkboxes.

Choosing 'sort options by Syntax element' from the drop down list will sort the list of by syntax conditions rather than by element.

Blank Lines

The Blank Lines tab allows you to set the number of blank lines to be created in various conditions.



Formatter - Blank Lines tab

Enter the number of blank lines (between 0-32) to be created in the following conditions:

Blank Lines

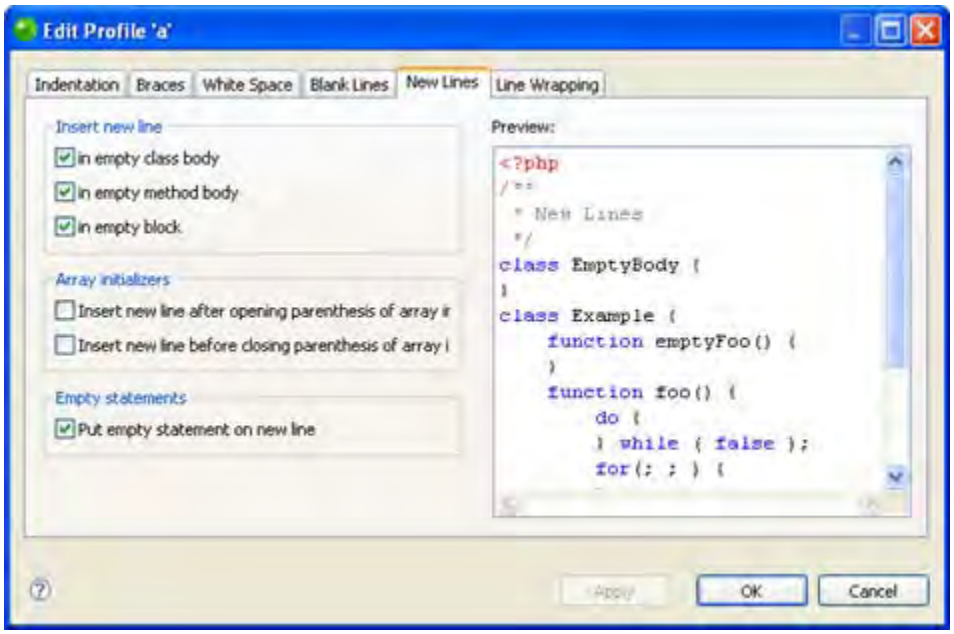
- Before class declarations
- Before constant declarations
- Before field declarations
- Before function/method declarations
- At beginning of function / method / body

Existing Blank Lines

- Number of empty lines to preserve

New Lines

The New Lines tab allows you to select when new lines will be created.



The New Lines tab contains the following options:

Insert New Line

Select whether to insert a new line in the following conditions:

- In empty class body
- In empty method body
- In empty block

Array Initializers

Select whether to:

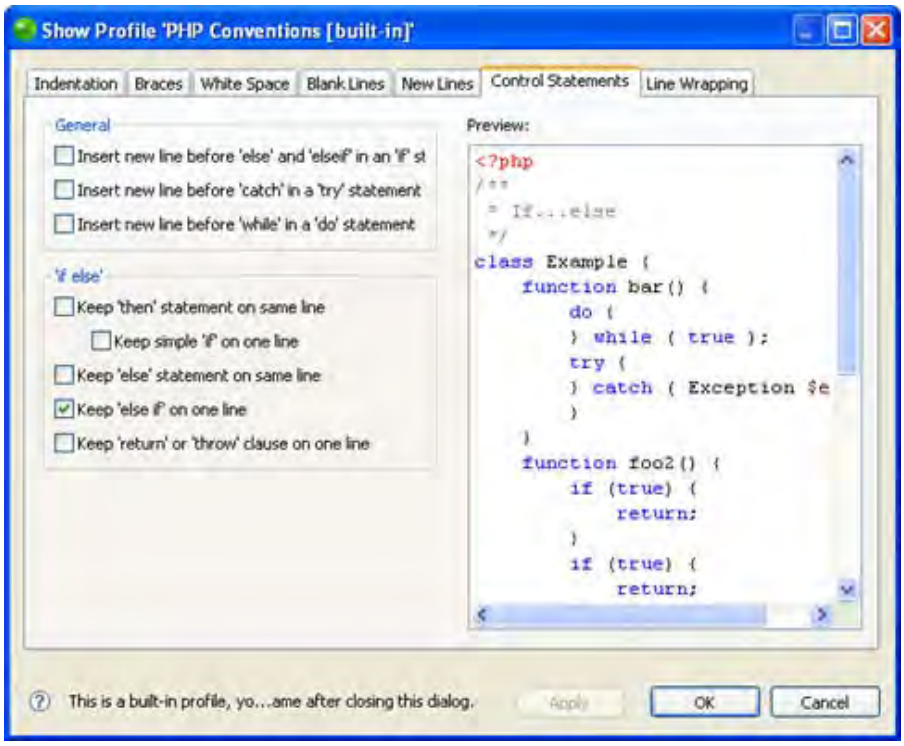
- Insert new line after opening parenthesis of array initializer
- Insert new line before closing parenthesis of array initializer

Empty statements

Select whether to put empty statements on a new line.

Control Statements

The Control Statements tab allows you to configure the line formatting for Control Statements.



The Control Statements tab contains the following options:

General

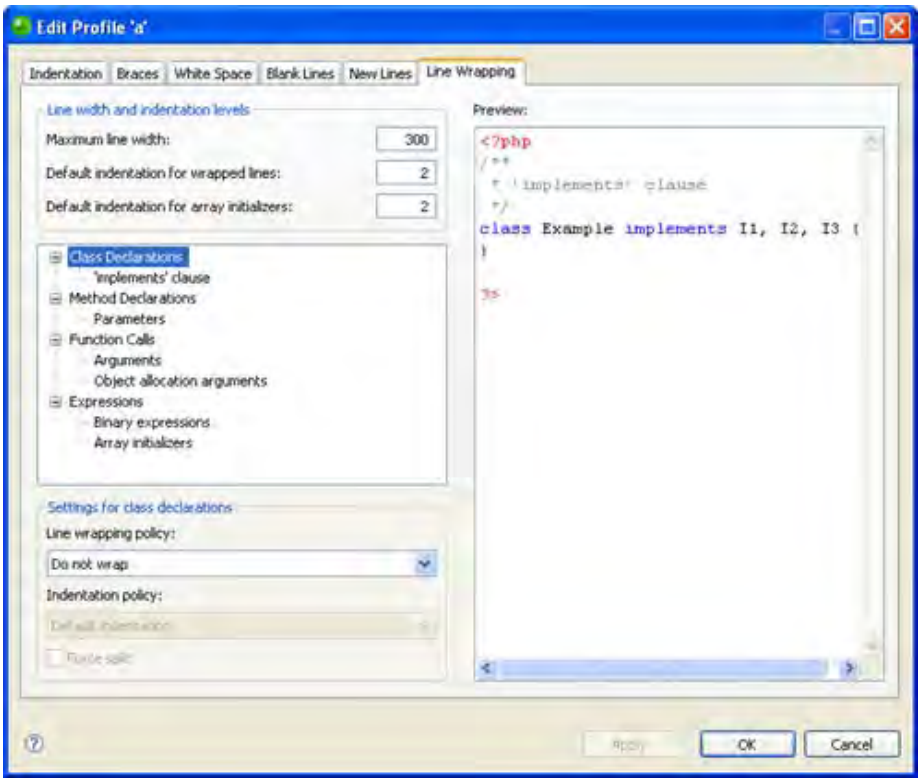
- Insert new line before 'else' and 'elseif' in an 'if' statement
- Insert new line before 'catch' in a 'try' statement
- Insert new line before 'while' in a 'do' statement

'if else'

- Keep 'then' statement on same line
- Keep simple 'if' on one line
- Keep 'else' statement on same line
- Keep 'else if' on one line
- Keep 'return' or 'throw' clause on one line

Line Wrapping

The Line Wrapping tab allows you to configure Line Wrapping properties.



The Line Wrapping tab contains the following options:

Line Width and Indentation Levels

- Maximum Line Width - Enter the maximum line width (in characters).
- Default indentation for wrapped lines - Enter the default indentation for wrapped lines (in characters).
- Default indentation for array initializers - Enter the default indentation for array initializers (in characters).

Line Wrapping and Indentation Policies

To set line wrapping and indentation wrapping policies for a specific element, select an element from the collapsible list and select an option from the 'Line wrapping policy' and 'Indentation policy' drop-down lists.
The elements for which the line and indentation policies can be applied are :

- Class Declarations
- 'implements' clause'
- Method Declarations
- Parameters
- Function Calls
- Arguments
- Object allocation arguments
- Expressions
- Binary expressions
- Array Initializers

To import an existing formatting configuration:

1. Click Import.
2. Select an XML file with the required configuration settings.
3. Click OK.

The new configuration will be added to the list.

To export a configuration file to an XML file:

1. Select the required configuration from the drop-down list.
2. Enter a name and location for the file.
3. Click OK.
4. An XML file will be created with the required settings.

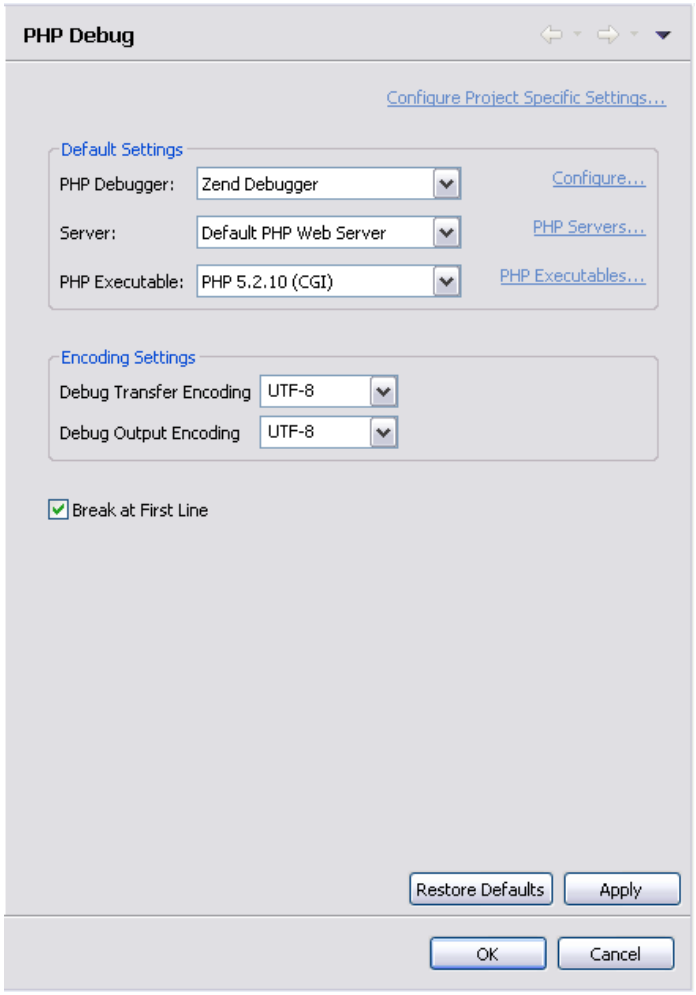
To apply Formatter settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Formatter Properties dialog will appear.
3. Select the required settings and click Apply.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project.
   -Or- Click No for a rebuild to be performed only when Zend Studio is restarted.
   -Or- Click Cancel to cancel the operation.

Debug Preferences

The Debug preferences page allows you to configure default settings for the debugging process.

The Debug Preferences Preferences page is accessed from Window | Preferences | PHP | Debug Preferences.



The settings that can be configured from the debug preferences page are:

Default Settings

- PHP Debugger - The default debugger is the Zend Debugger. Go to the Installed Debuggers preferences page to configure Zend Debugger settings.
- Server - Choose which server the debugger will use by default. Click the "PHP Servers" category to be taken to the PHP Servers management page. For more on this, see PHP Servers.
- PHP Executable - Choose the required default PHP version. Click the "PHP Executables" category to be taken to the PHP Executables management page. For more on this, see PHP Executables.

Encoding Settings

- Debug Transfer Encoding - Select the required debug transfer encoding from the drop-down list.
- Debug Output Encoding - Select the required debug output encoding from the drop-down list.
- Break at First Line - Mark this checkbox to force the debugging process to stop at the first line of code by default.

Note:

Further PHP encoding options can be accessed from the preferences menu under General | Content Types | Text | PHP Content Type.

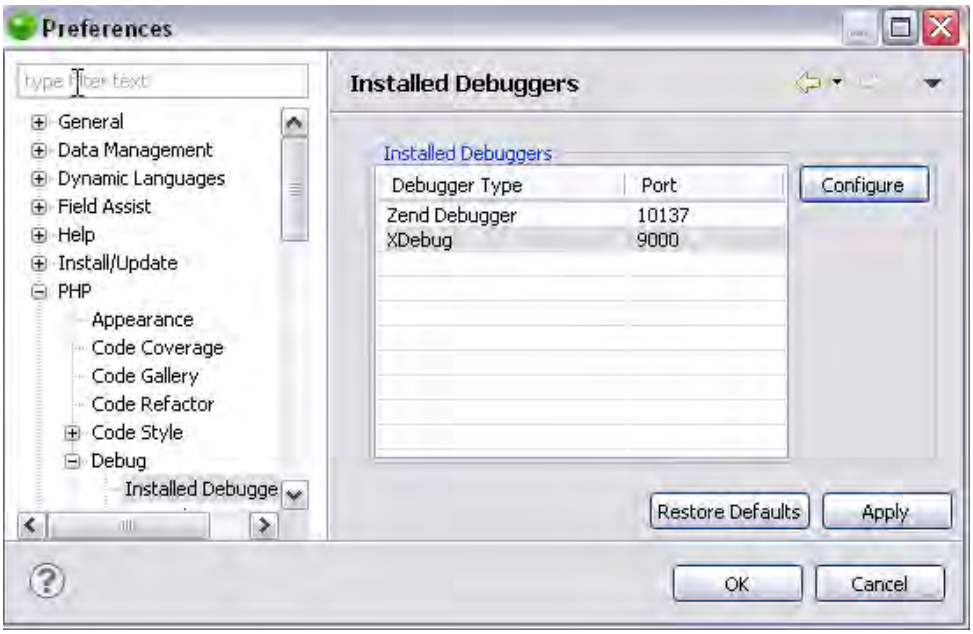
To apply Debug Preferences settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Debug Preferences Properties dialog will appear.
3. Select the required settings and click Apply.
4. A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
5. Click Yes to rebuild the project. Click No for a rebuild to be performed only when Zend Studio is restarted. Click Cancel to cancel the operation.

Installed Debuggers

The Installed Debuggers preferences page allows you to configure your Debugger settings.

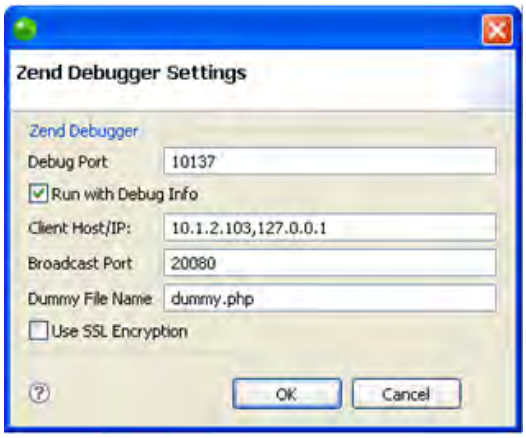
The Installed Debuggers Preferences page is accessed from Window | Preferences | PHP | Debug | Installed Debuggers.



To configure your Zend Debugger settings:

1. Select the Zend Debugger.
2. Click Configure.

   The Zend Debugger Settings dialog will open.



3. Configure the following:
   - Debug Port - The port which the Zend Debugger will use. The default port is 10137.
   - Run with Debug info - Mark the checkbox for Debug info, such as the Console view and the Browser Output, to be displayed when a Run configuration is executed.
   - Client Host/IP - Enter the Client Host/IP to which debugging results will be returned. Zend Studio will automatically search for and recognize the Client Host/IP, but entering a specific Host/IP will speed up the debugging process and decrease the likelihood of session timeouts.

Note:

If the field contains too many host/IPs, the session could timeout before a debugging connection is established. Entering a non-existent host/IP could cause the session to terminate.

   - Broadcast Port - The Broadcast Port allows your Zend Browser Toolbar or your Zend Server to detect your debugging preferences. The Broadcast Port number entered here must match the Broadcast Port entered in your Zend Browser Toolbar/Zend Platform.
     The default port is 20080.
   - Dummy File - This is the file which the PHP Script debugger uses in order to start a PHP script debugging session on a specified server. The name should be left as the default dummy.php. However, if this is changed, ensure the change has also been made on your server.
     See Ensuring the Placement of dummy.php for more information.
   - Use SSL Encryption - Mark this checkbox to Encrypt Communication using SSL. Your server must support this option in order for it to be applicable.
4. Click OK to return to the Installed Debuggers Preferences page.
5. Click Apply to apply your settings.

Step Filtering Preferences

The Step Filtering Preferences page allows you to select certain resources/file patterns which will not be 'stepped into'/displayed during debugging.
This feature should be used when there are files which you do not want to inspect during debugging. This is especially useful if you have included large external resources (such as a framework or library).

To enable the step filtering feature, mark the 'Use Step Filters' checkbox and the add the required resources to the 'Defined step filters' list.

Note:
The Step Filters functionality can be toggled on/off during debugging by clicking the Use Step Filters button on the Debug view toolbar.

To add a file name pattern to exclude:

1. Click Add Filter.
   The Add Step Filter dialog is displayed.
2. Enter the pattern to filter.

   Note:
   Wild cards are enabled, so entering *index.php will exclude all resource paths ending in index.php.

3. Click OK.

The file exclusion pattern will be added to the step filters list.

To add a resource to exclude:

1. Click Add Resource.

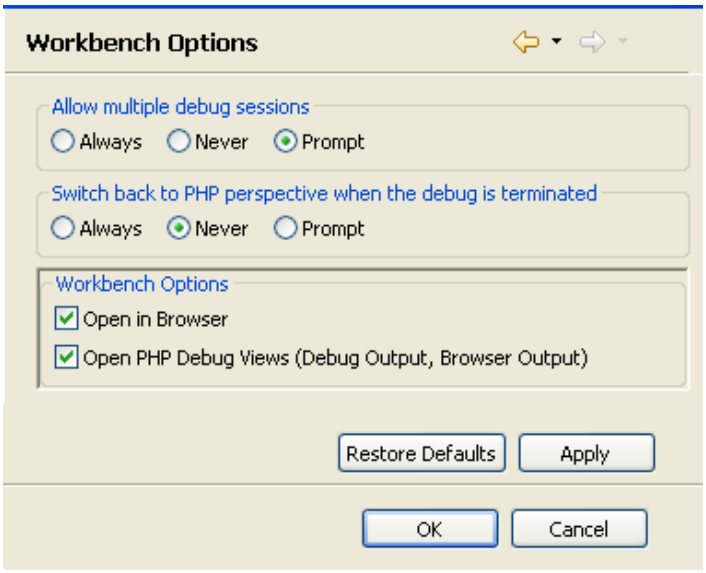
   The Select Resource dialog is displayed.
2. Select the resources (projects/folders/files) which you want to exclude and click OK.

The selected resources will be added to the step filters list.

Workbench Options Preferences

The Workbench Options preferences dialog allows you to configure the default behavior of the workspace during the debugging process.

The Workbench Options Preferences page is accessed from Window | Preferences | PHP | Debug | Workbench Options.



The Workbench Options configuration options are:

- Allow multiple debug sessions - Select whether to Allow multiple debug sessions to run simultaneously (Always, Never or Prompt).
- Switch back to PHP perspective when the debug is terminated - Select whether the PHP Persepctive will open when the debug is terminated (Always, Never or Prompt).

Workbench Options

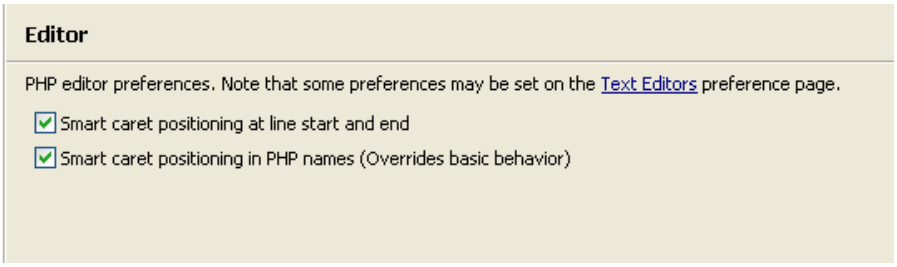
- Open in Browser - Mark the checkbox for the debugged files to be displayed in a browser during debugging.
- Open PHP Debug Views - Mark the checkbox for PHP Debug Views to be displayed when a debug session is launched.

  By default, a dialog will appear when a debug session is launched asking whether you want to open the Debug Perspective when a debugging session is run. To change this behavior, open the Perspectives Preferences dialog by going to Window | Preferences | Run/Debug | Perspectives and select Always, Never or Prompt in the 'Open the associated perspective when launching' category.

Editor Preferences

The Editor preferences page allows you to configure smart caret positioning behavior in the editor. Smart caret positioning determines where the cursor will jump to when certain positioning keys (e.g. Home / End) are pressed.

The Editor Preferences page is accessed from Window | Preferences | PHP | Editor.


Editor
PHP editor preferences. Note that some preferences may be set on the Text Editors preference page.
Smart caret positioning at line start and end
Smart caret positioning in PHP names (Overrides basic behavior)

To configure Smart Caret Positioning Preferences

1. Mark the required checkboxes to configure the following options:
   - Smart caret positioning at line start and end - If this checkbox is unmarked, the cursor will jump to the beginning/end of a line when Home/End are pressed. If it is marked, the cursor will jump to the beginning/end of the *typed* line (i.e. ignoring the tabs at the beginning/end of a line).
   - Smart caret positioning in PHP names - If this checkbox is marked, the cursor will jump to the beginning/end of a 'word' within a PHP element (class / function / variable) when Ctrl + left arrow/right arrow are pressed. When unmarked, it will jump to the beginning/end of the whole element.
2. Click Apply to apply your settings.

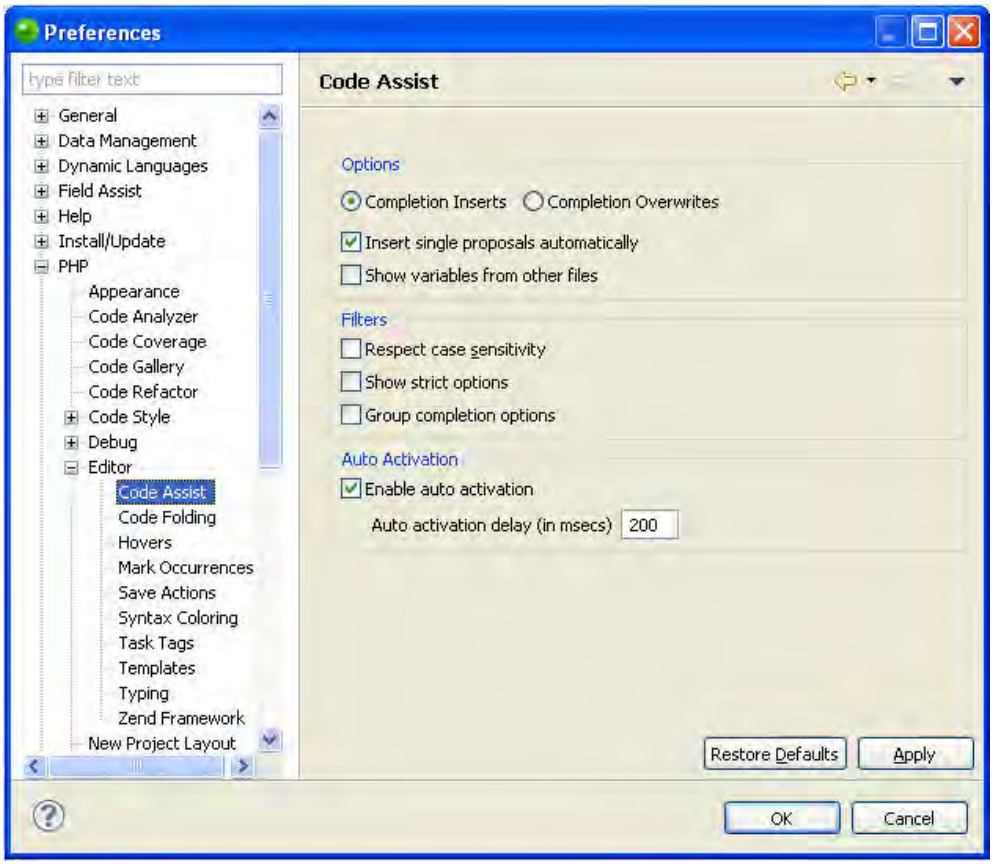
Note:
More editor settings can be accessed by clicking the "Text Editors" link.

Content Assist Preferences

The Content Assist feature enables the selection and insertion of existing code elements to complete partially entered code.

The Content Assist preferences page allows you to configure your Content Assist preferences.

The Content Assist Preferences Preferences page is accessed from **Window | Preferences | PHP | Editor | Content Assist Preferences**.



To configure Content Assist preferences:

1. Mark the required checkboxes to configure the following options:

Options

- Completion Inserts / Completion Overwrites - Select whether selecting an item from the Content Assist list will cause new code to be entered or existing code to be overwritten.
- Insert single proposals automatically - When only one content assist suggestion exists, the content assist suggestion will be inserted automatically.
- Show variables from other files - Shows variables which are in other files in the project.

Filters

- Respect case sensitivity – Define statements are used to create global constants. One can ”r;define(’r;MY_CONST’, 5)”, and then use ”r;MY_CONST” all over the application, this property is used to determine if ”r;my_const” will also be suggested as usage of that constant.
- Show strict options – PHP alerts users if they access static methods and fields from within an instance. Users can enable/disable this in the php.ini file for their php copy and hence can disable the content assist for these cases.

Auto-activation

- Enable auto activation - The content assist list will automatically be displayed.

Note:

If this is unmarked, you can still display the content assist list by pressing Ctrl+Space.

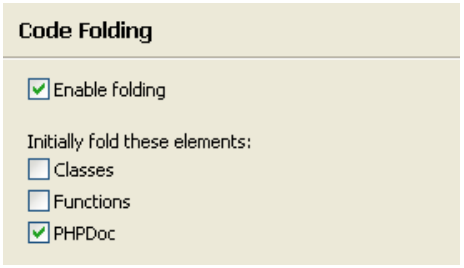
- Auto activation delay: Determines the delay before the Content Assist box is automatically displayed (in msecs).

2. Click Apply to apply your settings.

Code Folding Preferences

Code Folding enables you to 'collapse', or hide, certain sections of code while you are not working on them. This enables you to manage larger amounts of code within one window.
The Code Folding preferences page allows you enable / disable code folding and to select which elements should be folded by default.

The Code Folding Preferences page is accessed from Window | Preferences | PHP | Editor | Code Folding.



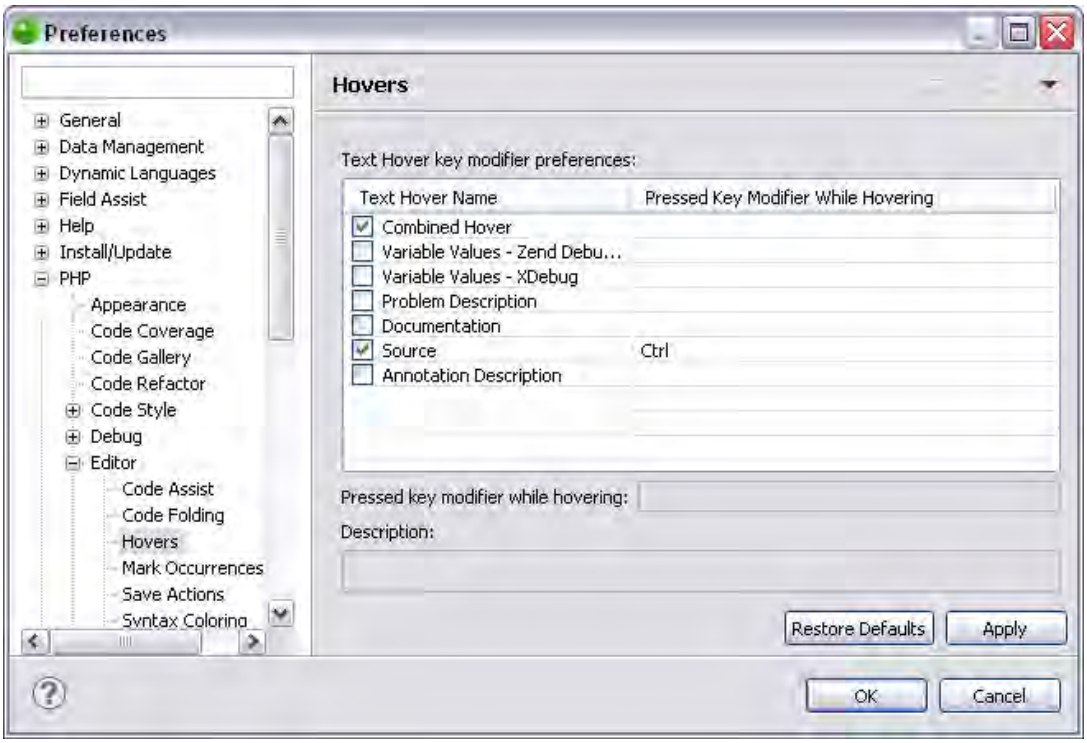
To configure your code folding preferences:

1. Mark the Enable folding checkbox to enable code to be folded.
2. Select which off the following elements should be folded by default by marking the relevant checkboxes:
   - Classes
   - Functions
   - PHPDocs
3. Click Apply to apply your settings.

Hovers Preferences

The Hover functionality will display information about an item when the mouse is placed on it. The Hovers preferences page allows you to configure the settings and shortcuts for the Hover functionality.

The Hovers Preferences Preferences page is accessed from Window | Preferences | PHP | Editor | Hovers Preferences.



The Text Hover key modifier preferences table allows you to modify hover key preferences for certain elements. Pressing the configured key while hovering over the element in the editor will display the relevant information or take the relevant action.
For example, applying the settings displayed in the screenshot above (Source key preference = Ctrl) and pressing Ctrl while hovering over an element in the editor will take you to that element's source.
You can configure key preferences for the following elements:

- Combined Hover - Tries the hover in the sequence listed in the table and uses the one which fits best for the selected element and the current context.
- Variable Values - Zend Debugger - Shows the value of the selected variable while debugging with the Zend Debugger.

- Variable Values - XDebug - Shows the value of the selected variable while debugging with XDebug.
- Documentation - Shows the documentation of the selected element.
- Problem Description - Shows the description of the selected problem.
- Source - Shows the source of the selected element.
- Annotation Description - Shows the description of the selected annotation.

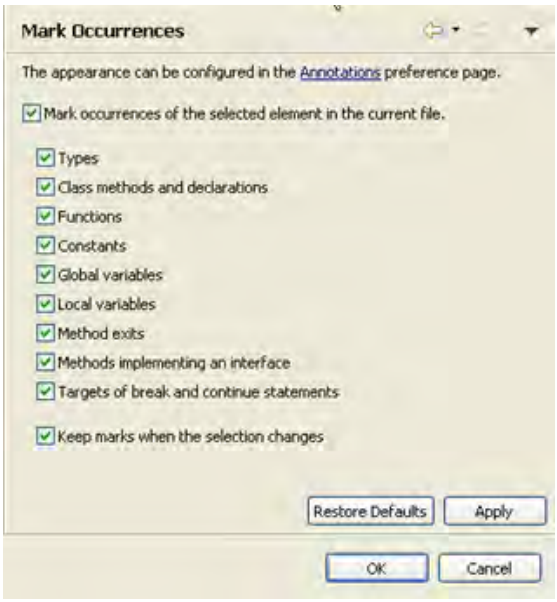
To configure the key preferences:

1. Mark the checkbox next to the required preference.
2. Enter the required key in the 'pressed key modifier while hovering' box.
3. Click Apply to apply your settings.

Mark Occurrences Preferences

The Mark Occurrences Preferences page allows you to configure for which types of elements the Mark Occurrences feature will be enabled.

The Mark Occurrences Preferences page is accessed from Window | Preferences | PHP | Editor | Mark Occurrences.



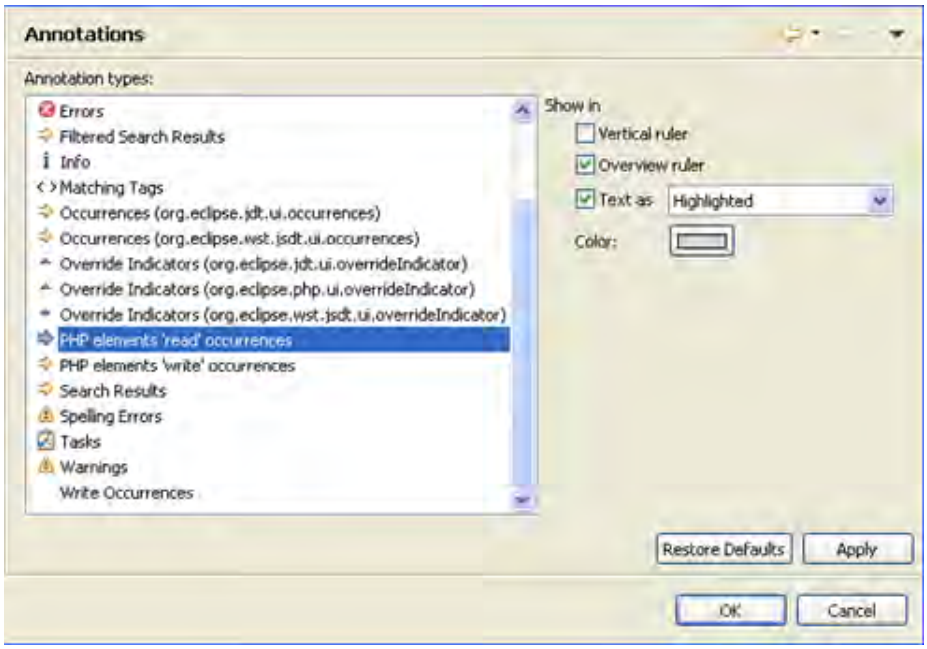
To configure Mark Occurrences preferences:

1. Mark the 'Mark occurrences of the selected element in the current file' checkbox to enable the Mark Occurrences functionality.
2. Configure the elements for which the Mark Occurrences feature will be enabled by marking the relevant checkboxes. The options are:
   - Types
   - Class Methods and declarations
   - Functions
   - Constants
   - Global variables
   - Local variables
   - Expressions throwing a declared exception
   - Method exits - Marks the exit points (throws / return / end of flow) of a method
   - Methods implementing an interface

- Targets of break and continue statements - Marks the scope (for, foreach, while, do-while or switch structure) of a break / continue statements
- HTML Tags

3. Mark the 'Keep marks when the selection changes' checkbox for marks to continue to be displayed once the cursor has been moved from the selected element.
4. Click Apply to apply your changes.

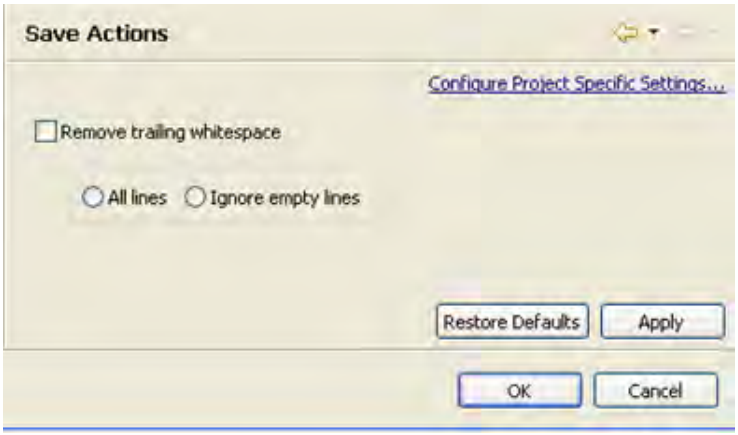
The appearance of the Mark Occurrences annotations which appear in the vertical ruler (to the left of the editor) and the annotation bar (to the right of the editor) can be configured in the Annotations preferences page (Window | Preferences | General | Editors | Text Editors | Annotations) by selecting PHP elements 'read' or 'write' occurrences.



Save Actions Preferences

The Save Actions Preferences page lets you remove trailing whitespace from a file each time you save it.

The Save Actions Preferences page is accessed from Window | Preferences | PHP | Editor | Save Actions.



To configure Save Actions:

1. Mark the 'remove trailing whitespace' checkbox to enable the removal of whitespace after every save.
2. Select whether All whitespace lines will be removed (by marking 'All lines'), or whether empty lines will be ignored (by marking 'Ignore empty lines).
3. Click Apply to Apply your settings.

Every time you save a file, whitespace will be removed according to the configured settings.

To apply Save Actions settings to a specific project only:

1. Click the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Save Actions Properties page will appear.
3. Select the required settings and click Apply.

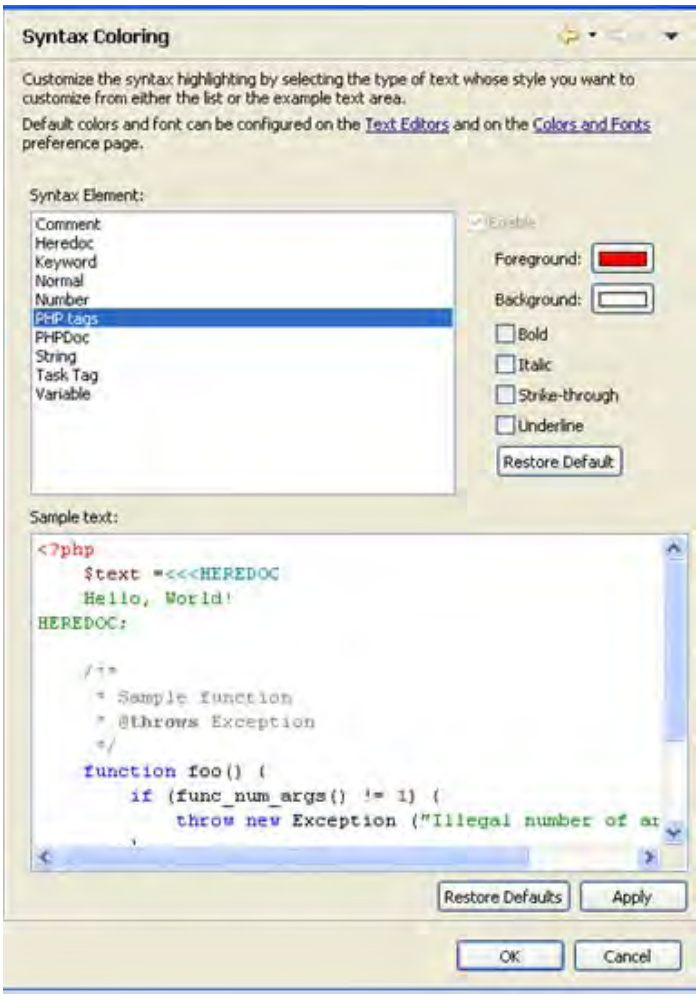
Note:

Save Action settings can also be configured for an existing project by right-clicking the project in PHP Explorer view and selecting Properties | Save Actions.

Syntax Coloring Preferences

The Syntax Coloring preferences page allows you to set the foreground color, background color and font type for different icons, in order to make your script manageable and easier to read.

The Syntax Coloring Preferences page is accessed from Window | Preferences | PHP | Editor | Syntax Coloring.



To configure the colors and fonts for an item:

1. Select the required item from the Syntax element list.
2. Click on Foreground or Background to select a colour.
3. Select what formatting, if any, you would like to apply to the text (Bold, Italic, Strikethrough, Underline)
4. Click Apply to apply your settings.

The Sample text box displays a preview of the different elements.

More color and font options can be configured by opening the preferences page, accessed from Window | Preferences, and selecting:

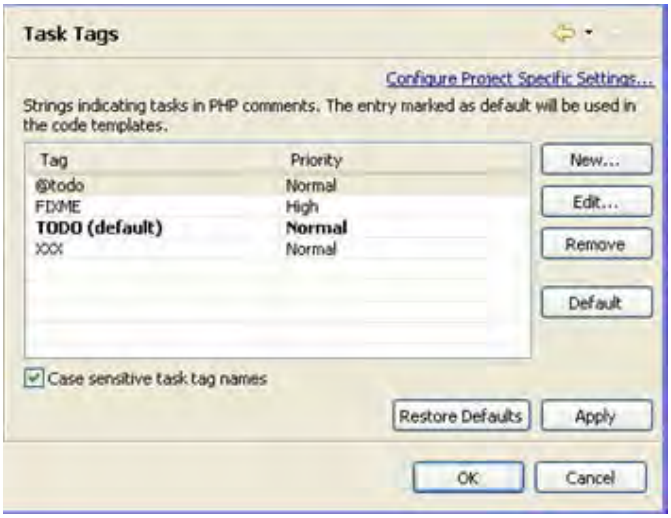
- General | Appearance | Colors and Fonts
- General | Editors | Text Editors | Annotation
- General | Editors | Text Editors | Quick Diff
- Run / Debug
- Run / Debug | Console
- Team | CVS | Console

Task Tags Preferences

The Task Tags preferences page allows you to add new task tags and edit existing ones. Tasks are used as reminders of actions, work to do or any other action required by the programmer. Task tags are strings used by Zend Studio to recognize when tasks are added in your script. Anything after these strings inside a comment will be recognized as a task.
See the 'Tasks view' topic in the Workbench User Guide for more information on using tasks.

Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

The Task Tags Preferences page is accessed from Window | Preferences | PHP | Editor | Task Tags.



Four common strings are included in the list by default.

To add a new Task Tag:

1. Click New.
2. Enter a tag name and priority (High/Normal/Low). Tags may contain any character string.
3. Click OK.
4. Click Apply to apply your settings.

The new tag will be added to the list and will trigger a task when inserted in the editor.

To edit a tag:

1. Double click the tag -or- select it and click Edit.
2. Edit the tag name or priority.
3. Click OK.
4. Click Apply to apply your settings.

Selecting a tag and clicking Default will set the task tag as the default one to be used in Code templates. See the Templates Preferences topic for more on template preferences.

Note:

If the 'case sensitive task tag names' checkbox is marked, task tag names will be case sensitive.

To apply Task Tags settings to a specific project only:

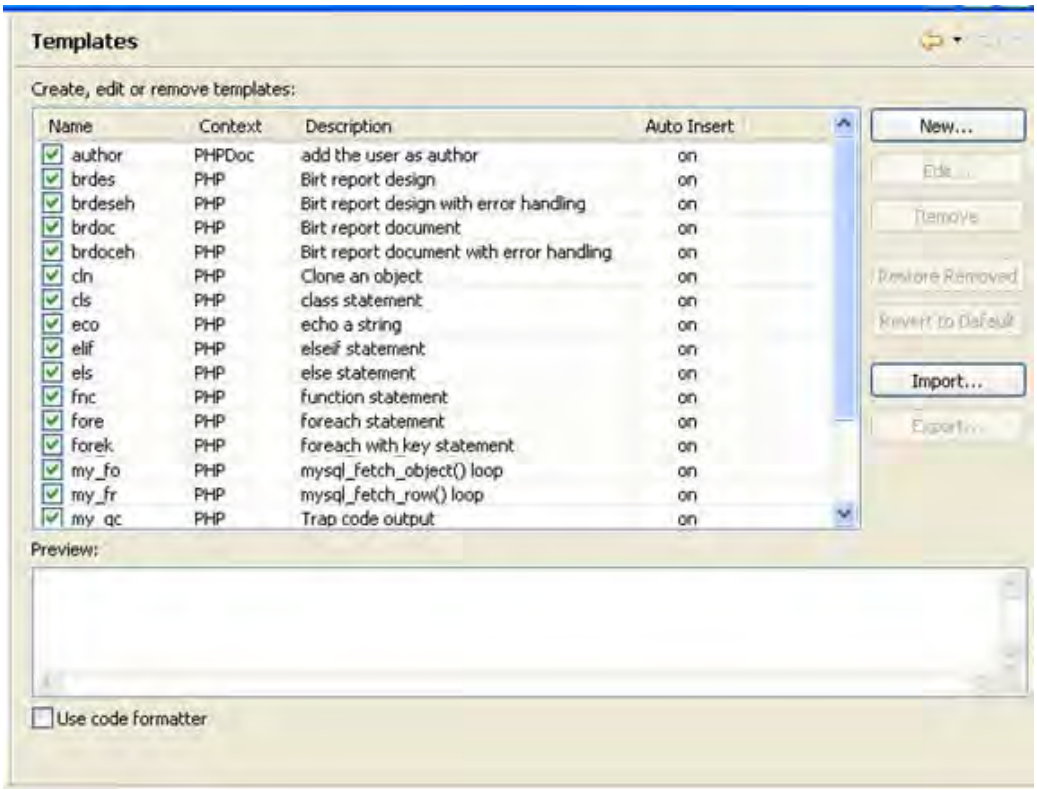
1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Task Tags Properties dialog will appear.
3. Select the required settings and click Apply.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project
   -Or- Click No for a rebuild to be performed only when Zend Studio is restarted.
   -Or - Click Cancel to cancel the operation.

Templates Preferences

The Templates Preferences page allows you to create, edit, delete, import and export templates. Templates are shortcuts used to insert a pre-defined framework of code. The purpose is to save time and reduce the potential for errors in standard, repetitive code units. Once a template is inserted, you can complete the code quickly using manual and automated code entry methods.

To insert a template into your code, enter the first few letters of its name and press Ctrl+Space to activate the Content Assist function and open the content assist list. Select the required template from the list. Templates are marked with a icon. For more on using the content assist function, see Working with Content Assist.

The Templates Preferences page is accessed from **Window | Preferences | PHP | Templates**.


Templates
Create, edit or remove templates:
Name Context Description Auto Insert
author PHPDoc add the user as author on
brdes PHP Birt report design on
brdeseh PHP Birt report design with error handling on
brdoc PHP Birt report document on
brdoceh PHP Birt report document with error handling on
cln PHP Clone an object on
cls PHP class statement on
eco PHP echo a string on
elif PHP elseif statement on
els PHP else statement on
fnc PHP function statement on
fore PHP foreach statement on
forek PHP foreach with key statement on
my_fo PHP mysql_fetch_object() loop on
my_fr PHP mysql_fetch_row() loop on
my_qc PHP Trap code output on
New...
Edit...
Remove
Restore Removed
Revert to Default
Import...
Export...
Preview:
Use code formatter

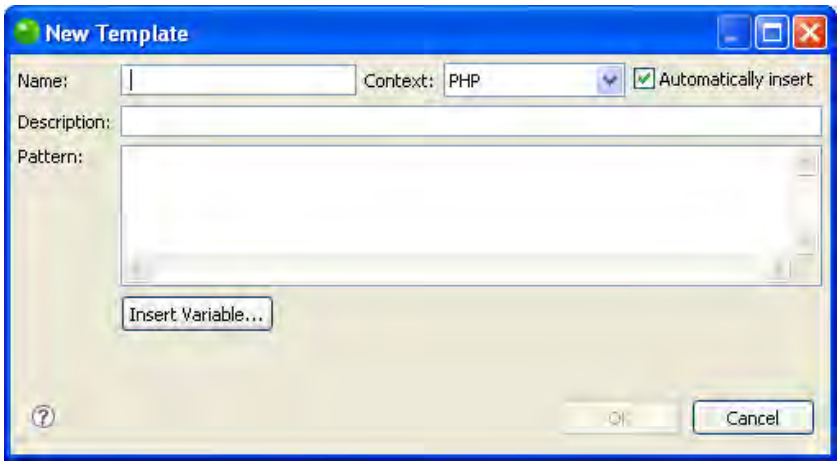
To remove a template from the list of available options, unmark its checkbox in the list.
To edit an existing template, select it from the list and click Edit.

Creating a new Template

This procedure describes how to create a new template to be added to the template list.

To create a new template:

1. Click New.
2. The New Template dialog will open.



3. Enter the template's details:
   - Name - A short name to identify the template (e.g. 'while' for a template for a while loop).
   - Context - The code context when the template will be available (PHP, PHPDoc, New PHP, BIRT or Zend Framework). e.g. PHP templates will only be available for use when writing PHP code.
   - Description - A short description of the template's code.
   - Pattern - The pattern is the actual code string that will be inserted into the editor whenever this template is selected.
     Use the Insert Variable button to select from a list of common variables.
4. Click OK.

Your template will be added to the list and will be available from the Content Assist in the relevant context.

Exporting and Importing Templates

Zend Studio enables you to export and import Templates, created within XML files in the following format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<templates>
<template autoinsert="true" context="php" deleted="false"
description="description" enabled="true" name="for">
for($$i = 0; $$i < 1; $$i++){ }
</template>
```

To import a template:

1. Click Import to open the Import Template's browser.
2. Select the location to import the relevant XML file containing the template information.
3. Click Open.

The templates contained in the template.xml file will be imported into the list of Templates.

To export a template:

1. Select the template(s) for export from the Template list.
2. Click Export to open the Export Template's dialog.
3. Select the location to save the XML file to.
4. Click Save.

An XML file will be created with the template information.

Note:

If you selected more than one template to export, all of them will be present in the exported XML file. Each of the original Templates is bounded by: < template > </template>

Typing Preferences

The Typing preferences page allows you to configure the code and language patterns that Zend Studio will automatically complete, and whether the tab key will indent the current line.

The Typing Preferences page is accessed from Window | Preferences | PHP | Editor | Typing.

Automatically Close

Zend Studio can be set to automatically complete the following types of patterns:

- "Strings" - A pair of double quotes ("") will be inserted when a single quotation mark (") is entered.
- (Parentheses) and [Square] brackets - A pair of brackets will be inserted when the opening bracket is entered.
- {Braces} - A pair of braces will be inserted when the opening brace is entered.
- PhpDoc and comment regions - Automatically creates phpDoc Blocks
- Add phpDoc tags - Adds phpDoc tags within the phpDoc Block comment.
- Close PHP tag - A closing PHP tag (?>) will be inserted when the opening PHP tag (<?php) is entered.

To configure your 'automatic close' options:

1. Mark the checkboxes of the patterns you would like Zend Studio to auto-complete.
2. Click Apply to apply your settings.

To use the auto-complete function, type the opening character in the editor. The matching character will be automatically inserted.

Tabulators

To be able to indent a selected line in the editor using the Tab Key:

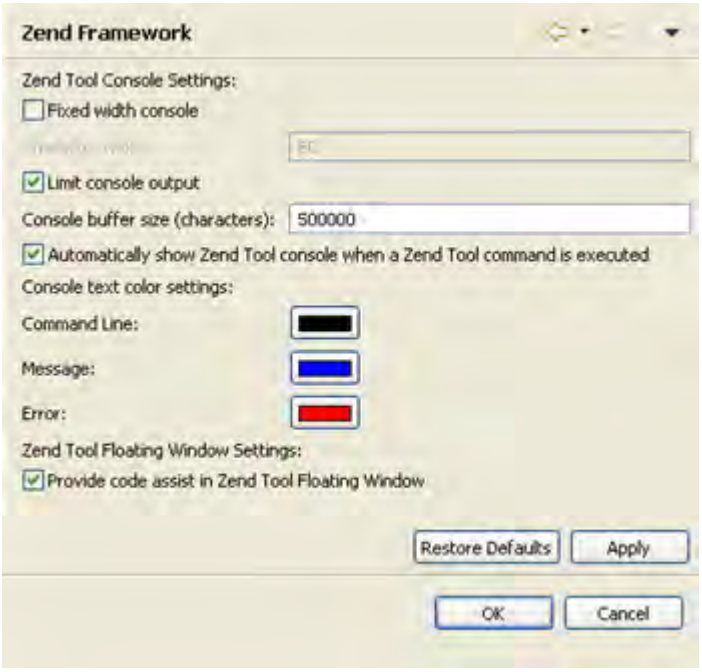
1. Mark the 'tab key indents the current line' checkbox.
2. Click Apply to apply your settings.

For more on indentation preferences, see the Formatter Preferences page.

Zend Framework Preferences

The Zend Framework preferences page allows you to configure the display of the Zend Tool Console.

The Zend Tool console displays messages and warnings when executing Zend Framework commands which utilize the Zend_Tool, such as create new Zend Framework Project or create new Zend Controller. See Using the Zend Tool Floating Window for more information.



The Zend Framework Preferences Preferences page is accessed from Window | Preferences | PHP | Editor | Zend Framework.

To configure the Zend Tool Console display settings:

1. Configure the following options:

   Zend Tool Consolse Settings:

   - Fixed width console - Mark the checbox to limit the width of messages displayed in the Zend Tool console.
     Enter the required width in characters. The value must be between 1000 and 2,147,483,646.
   - Limit console output - Mark the checkbox to limit the amount of characters stored and displayed in the Zend Tool Console.
     Enter the required size in characters. The value must be between 80 and 2,147,483,646.
   - Automatically show Zend Tool console when a Zend_Tool command is executed - Mark this checkbox for the Zend Tool Console to be automatically displayed whenever a command which utilizes the Zend_Tool (e.g. create new Zend Framework project) is executed.
   - Console text color settings - Edit the colors for text displayed in the Zend Tool Console for the following:
     - Command line
     - Messages
     - Errors displayed

   Zend Tool Floating Window Settings:

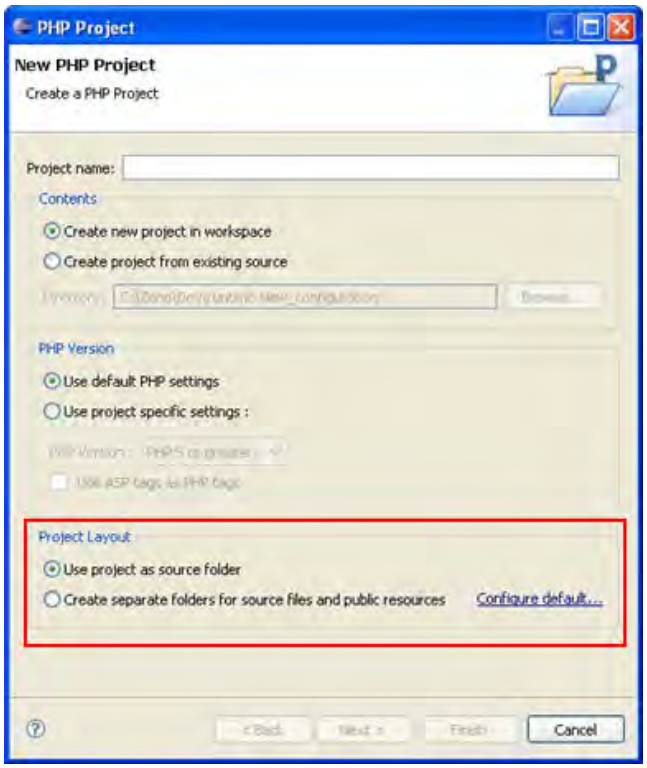
   - Provide content assist in Zend Tool Floating Window - Provides content assist options for Zend_Tool commands in the Zend Tool Floating Window.
     See Using the Zend Tool Floating Window for more information.

2. Click Apply to apply your settings.

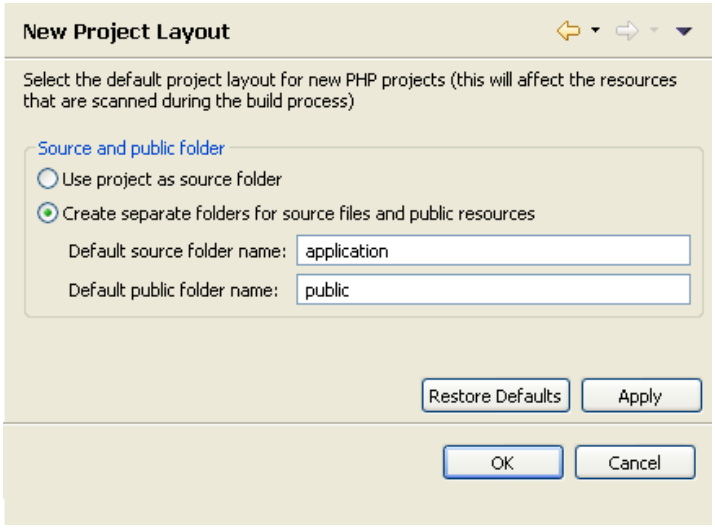
New Project Layout Preferences

The New Project Layout Preferences page allows you to configure the default layout for new PHP projects. The default layout will configure whether all folders under the project root will be considered as 'source' folders for the Build process, or whether separate folders will be created for resources which are included or excluded from the Build process.

The preferences configured in the New Project Layout Preferences page will affect the default option selected in the Project Layout category of the New PHP Project wizard. This setting can be modified per project during creation.



The New Project Layout Preferences page is accessed from Window | Preferences | PHP | New Project Layout Preferences.


New Project Layout
Select the default project layout for new PHP projects (this will affect the resources that are scanned during the build process)
Source and public folder
Use project as source folder
Create separate folders for source files and public resources
Default source folder name:
application
Default public folder name:
public
Restore Defaults
Apply
OK
Cancel

To configure the default New PHP Project Layout:

Select the required option:

- Use project as source folder - By default, all folders created under the project root will be considered 'source folders' and will be scanned during the Build process. See Configuring a Project's PHP Build Path to configure the project's PHP Build Path.
- Create separate folders for source files and public resources - Two folders will be created under your project root - one 'source' folder for resources which will be scanned during the Build process and one 'public' folder which will be skipped during the Build process.

  You can change the default names for these folders by entering the required name in the 'Default source/public folder name' fields.

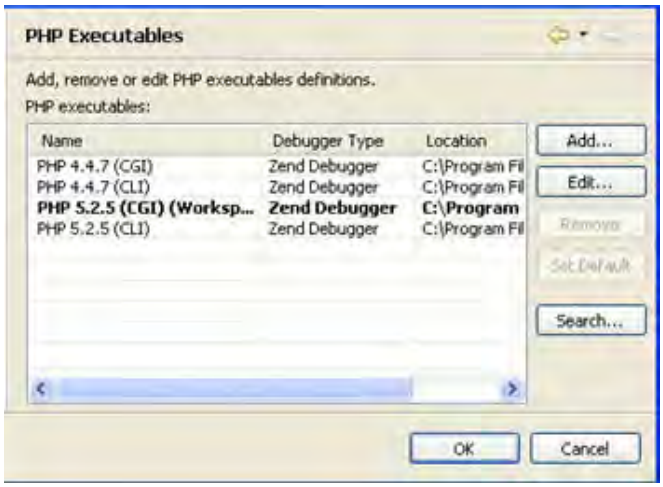
PHP Executables Preferences

The PHP Executables Preferences page allows you to add, edit, remove and find PHP Executables.

The internal debugger in the defined PHP Executable is used for local PHP Script debugging.

Zend Studio comes with a bundled pre-configured PHP Executable and local debugger.

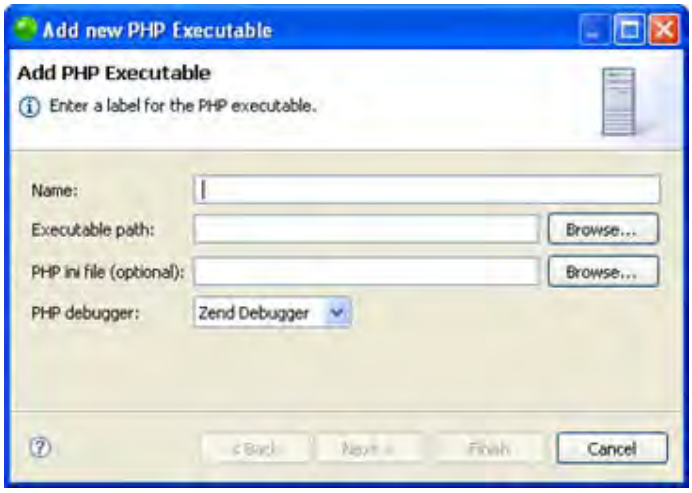
The PHP Executables Preferences Preferences page is accessed from Window | Preferences | PHP | PHP Executables Preferences.



To add a PHP executable to the list:

1. Click Add.

   The Add PHP Executable dialog will appear.



2. Enter a name for the PHP Executable.
3. In the Executable path selection, enter the location of the PHP executable on your

file system.

4. Select the PHP ini file to be associated with the PHP Executable by clicking Browse (optional).
5. Select the PHP Debugger to be used with the executable. By default this will be the Zend Debugger.
6. Click Finish.

The PHP executable will be added to your list.

To search for a PHP executable on your local file system:

1. Click Search.
2. In the Directory Selection dialog, select the folder to search.
3. Click OK. Zend Studio will search for PHP executables in the location specified.

Any found PHP executables will be added to the list.

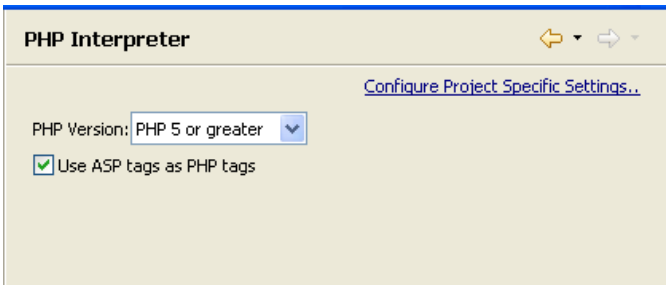
To change the name of the new PHP executable, select it from the list and click Edit.

If a PHP executable is not found in the specified directory, select a different location or add the PHP Executable manually by following the instructions under 'To add a PHP executable to the list', above.

PHP Interpreter Preferences

The PHP Interpreter preferences page allows you to set which PHP version to use for the project. This will affect the internal debugger, code analyzer and content assist.

The PHP Interpreter Preferences page is accessed from Window | Preferences | PHP | PHP Interpreter.

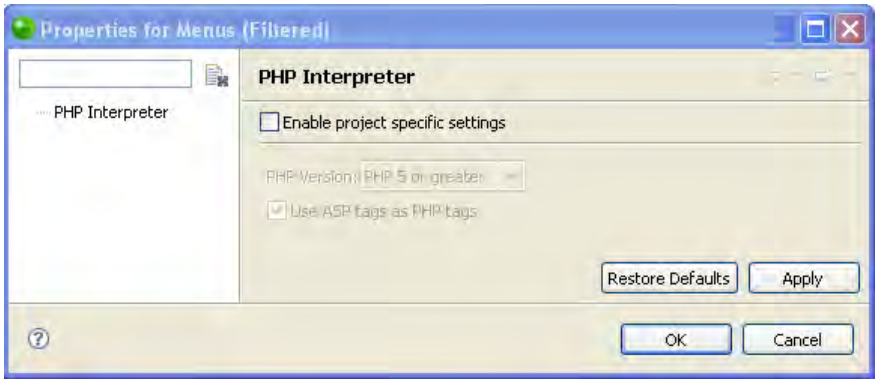


To configure your PHP version:

1. Select the PHP Version to use. The default options are PHP 4 or PHP 5 or greater. See PHP Support for more on effects the PHP version settings have.
2. Mark the "Use ASP tags as PHP tags" checkbox in order for Code Completion to respond to ASP tags in the same way as it responds to PHP tags.
3. Click OK to apply your settings.

To use a different PHP Interpreter for a specific project:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the specific project from the list.
3. Another PHP Interpreter preferences page will open.



4. Mark the Enable project specific settings checkbox.
5. Choose your PHP version.
6. Click Apply.
7. A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
8. Click Yes to rebuild the project. Error parsing will be performed according to the PHP version chosen.

   If you click No, the rebuild will be performed the next time Zend Studio is restarted.

PHP Libraries Preferences

The PHP Libraries Preferences page allows you to create and maintain an external code library. Enabling PHP Libraries in your project allows libraries to be referenced by the project and makes the elements within these resources available for operations such as Content Assist and Refactoring.

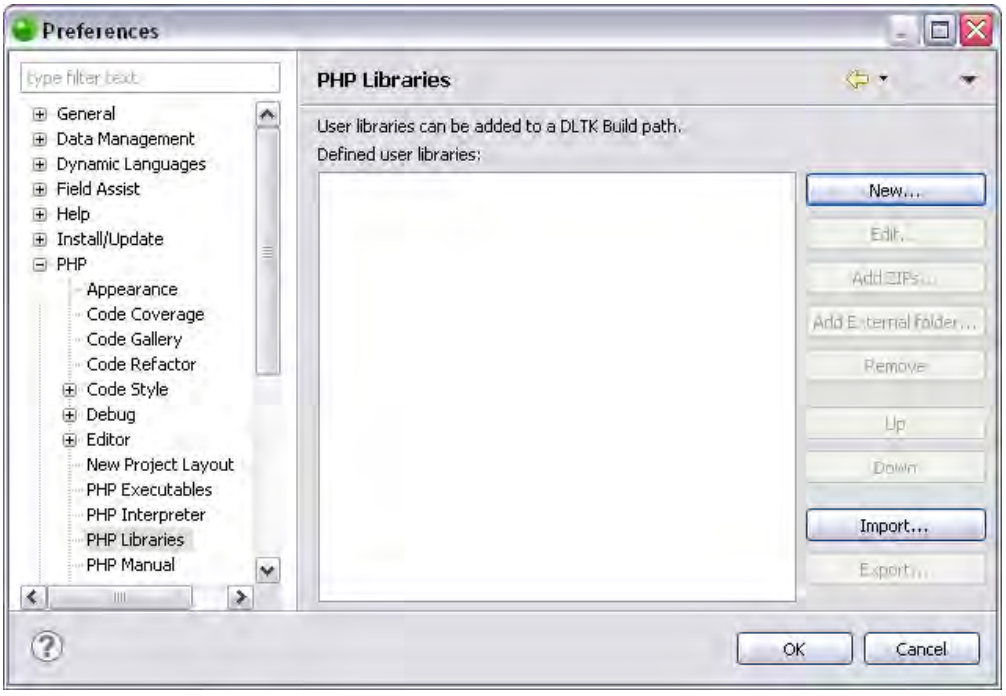
Note:

You may change the order your user libraries are in by using the **Up** and **Down** buttons. The order in which the libraries are arranged in this page defines the order they are available in Zend Studio's functionality, such as content assist.

The PHP Libraries Preferences page allows you to do the following:

- Add PHP Libraries
- Add External Folders to PHP Libraries
- Export PHP User Libraries
- Import PHP User Libraries
- Edit PHP User Libraries
- Edit PHP Library Components or Folders
- Remove a PHP Library or Library Folder

The PHP Libraries Preferences page is accessed from **Window** | **Preferences** | **PHP** | **PHP Libraries**.



Note:

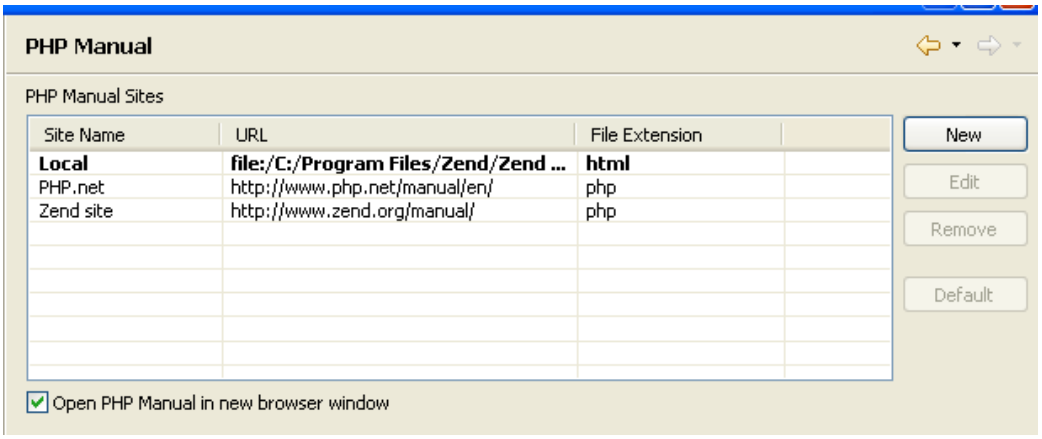
Once you have added a user library in the PHP preferences page, you must also add it to your PHP Include Path of the project in which you would like to have it available. For more information see Configuring a Project's PHP Include Path.

PHP Manual Preferences

The PHP Manual preferences page sets the location of sites which allow you to access and view the PHP Manual and allows you to add, edit or remove sites.

The PHP Manual contains an explanation of PHP functions. PHP Manuals can be accessed online or locally from within Zend Studio in order to provide an immediate explanation of the functionality and proper use of all PHP functions.

The PHP Manual Preferences Preferences page is accessed from Window | Preferences | PHP | PHP Manual Preferences.



To add additional sites to the list:

1. Click New.
2. Enter the Name of a site allowing access to the PHP Manual and its URL, Local Directory location or Windows CHM File location.
3. Choose whether its file extension is php, htm or html.
4. Click OK. Your new site will be added to the list.

See the PHP Manual Integration topic for more information.

Mark the 'Open PHP Manual in new browser window' checkbox to select that each request to open the manual will appear in a new browser tab of the Editor.

Note:

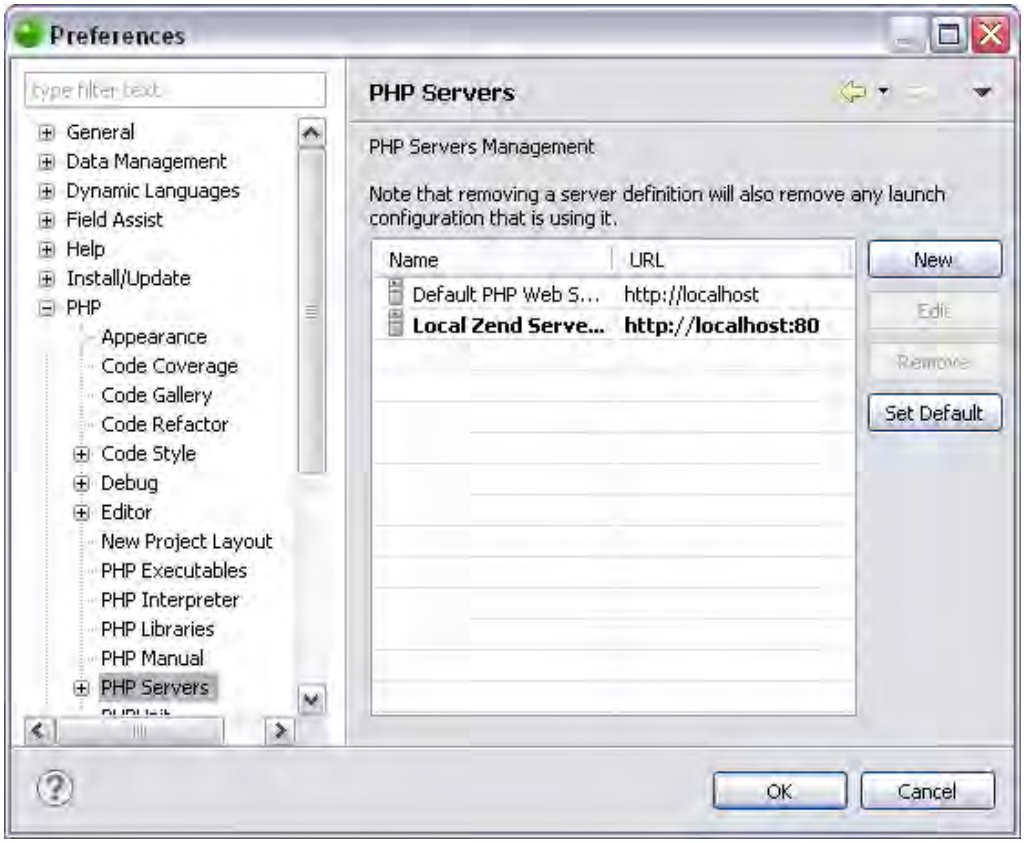
The initial, default site cannot be removed or edited.

Click OK to apply your settings.

PHP Servers Preferences

The PHP Servers Preferences page will display a list of your currently configured servers and allow you to add servers or edit settings for existing servers. The server settings will be used for debugging/profiling files on a remote server.

The PHP Servers Preferences page is accessed from **Window | Preferences | PHP | PHP Servers**.



One server configuration setting is configured by default and will point to the URL http://localhost.

Note:

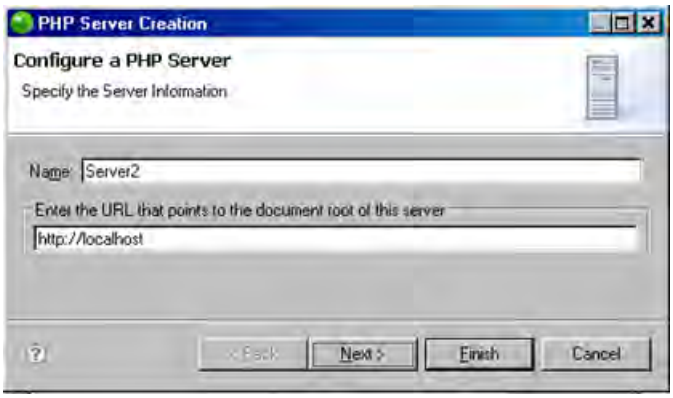
You can benefit from Zend's expertise by using the Zend Server, which includes the Zend Debugger, in order to perform optimal debugging and profiling functionalities. See Zend Server Integration for more information.

If you have a Zend Server installed on your local machine it is automatically detected and configured as the default server in the PHP Servers list. See Configuring Zend Server Settings in Zend Studio for more information.

To add a new server to the list or edit an existing server configuration:

1. Click New

   -Or- select an existing server and click Edit.



2. Enter the name of your server.
3. Enter the URL that points to its document root.
4. Click Next.

   The Server Path Mapping dialog appears.
5. If you would like to map a path on your server to a local path, click Add and enter:
   - The path on your server.
   - The path you would like to map it to in your Workspace or on your File System.

     See Adding a Server Location Path Map for more information.
6. Click OK and Next.
7. If you would like to enable integration with Zend Server, mark the 'Enable Zend Server Integration' checkbox and enter the relevant information:
   - URL Suffix - The suffix which should be added to the URL of your document root in order to browse to your Zend Server GUI.
   - Port number - Enter the port number you defined during Zend Server installation. See the Zend Server FAQ site for default port number settings.
   - Password - The password used to access your Zend Server GUI.
8. Click Next.
9. If you would like to enable Tunneling, mark the 'Enable Tunneling' checkbox and enter the relevant information:
   - Specify Return Host - If the debug session occurs on a node other than the central server, unmark the checkbox and specify the IP address of the central server to which the tunnel will be opened.
   - Automatically Connect on Startup
   - Send Authentication Information - User Name and Password.

See Setting Up Tunneling for more information.

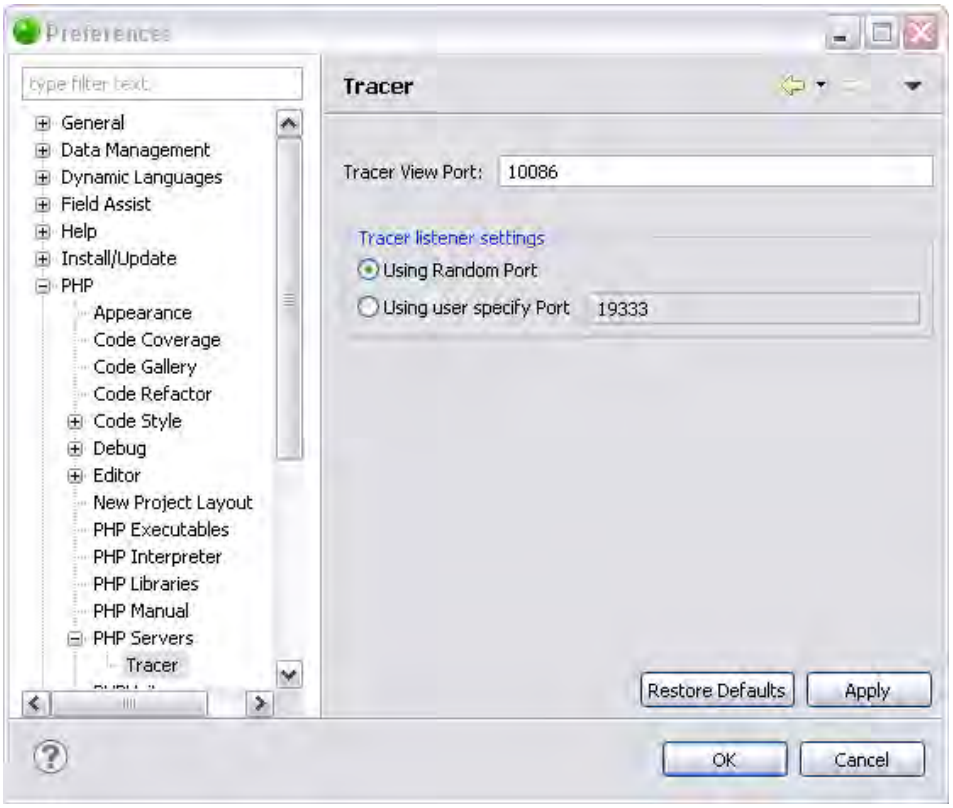
10. Click Finish.

Your new server configuration will be added to the server list and will be available for actions such as debugging, profiling and Zend Server integration.

Tracer Preferences

The Tracer Preferences page displays the ports that the Code Tracing functionality will use. The ports that Zend Studio chooses by default should not be changed unless they are specifically blocked or in use by another application.

The Tracer Preferences page is accessed from **Window | Preferences | PHP | PHP Servers | Tracer**.



Editing the Tracer Listener Settings

This procedure describes how to change the port the Code Tracing functionality will use. By default Zend Studio will find the first available port.

Important Note:

The ports in the Tracer preferences should only be changed if the default port is blocked or in use by another application.

To edit the tracer listening port:

1. Go to **Window | Preferences | PHP | PHP Servers | Tracer.**
   The Tracer preferences page opens with the "Use a random port" option selected by default. The random port that has been selected is expressed in the "Use a user specified port" text field.
2. Select the "Use a user specified port" option and enter the port number in the text field.
3. Click **OK** to save the changes.
   The Tracer listening port has been changed.

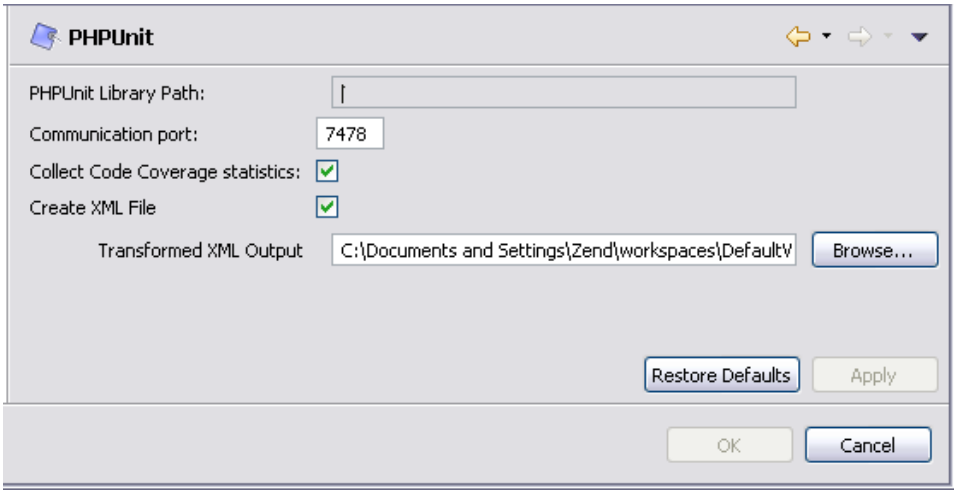
If you would like Zend Studio to revert back to a default port select the "Use a random port" option. To revert back to the original port that was used click **Restore Defaults**.

PHPUnit Preferences

The PHPUnit Preferences page allows you to see your PHPUnit Library Path and set the PHPUnit's communication port.

PHP Unit tests are a way of constantly testing your code to ensure the right output is being generated each time.
For more on PHP Unit Testing, see the PHPUnit Testing Tutorial.

The PHPUnit Preferences page is accessed from Window | Preferences | PHP | PHPUnit.



The PHPUnit Library Path displays the location of your PHPUnit Library. These settings cannot be changed.

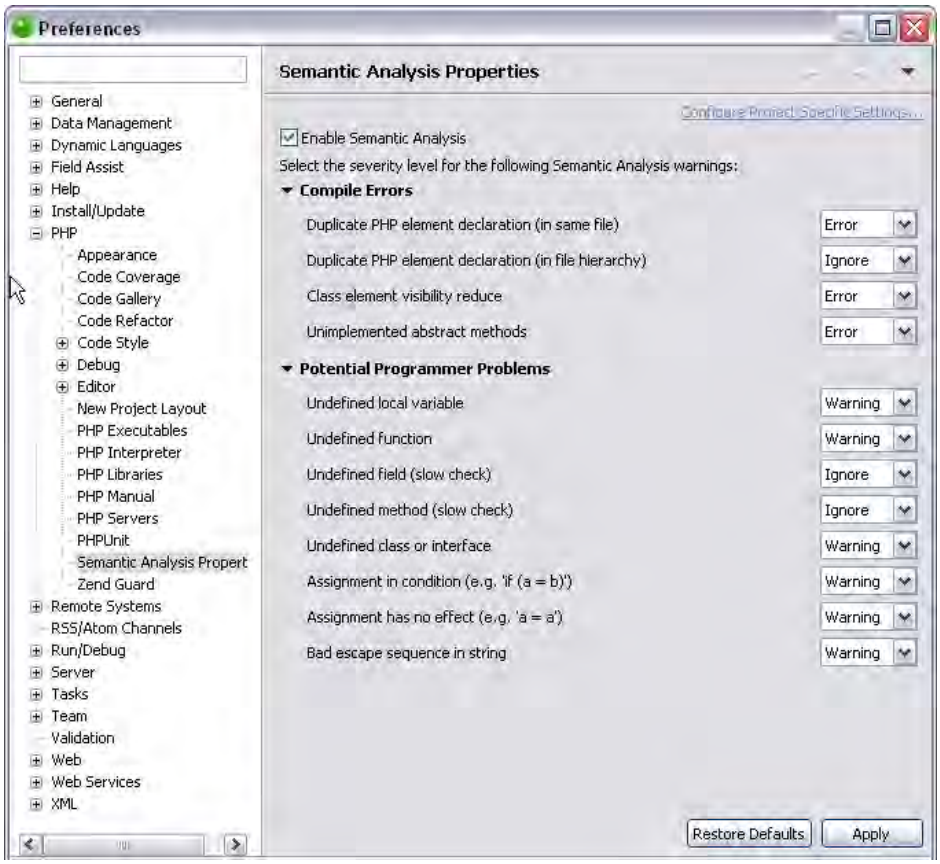
To configure your PHPUnit settings:

1. Set the communication port which will be used when PHPUnit Tests are run. The default port number is 7478. Ensure that the selected port is not already in use.
2. Mark the 'Collect Code Coverage statistics' checkbox to enable code coverage while running unit tests.
3. Mark the 'Create XML File' checkbox to enable XML file generation and click Browse in the Report Location option to select the location in which they will be created. XML files can later be generated into reports from the PHPUnit Test results view by clicking . See Reporting on PHPUnit Test Results for more information.
4. Click OK to apply your settings.

Semantic Analysis Preferences

The Semantic Analysis feature enables warning and error messages to be displayed when Zend Studio detects possible errors or problems in your script.

The Semantic Analysis Preferences page is accessed from **Window | Preferences | PHP | Editor | Semantic Analysis Properties**.



To enable and configure Semantic Analysis for all projects:

1. Mark the 'Enable Semantic Analysis' checkbox.
2. Select the severity level displayed in error messages (warning, ignore or error) for a variety of occurrences, divided into the following categories:
   - Compile Errors
   - Potential Programmer Problems

   To select a severity level for an event, click the arrow next to each header to display the possible options (if not displayed) and select the severity level from the drop-down list next to each option.
3. Click Apply to apply your settings.
   A prompt dialog is displayed indicating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project. Click No for a rebuild to be performed only when Zend Studio is restarted. Click Cancel to cancel the operation.

Compile Errors

Allows you to select the severity level (error, warning or ignore) for the following:

- Duplicate PHP element declaration (in same file)
- Duplicate PHP element declaration (in file hierarchy)
- Class element visibility reduce
- Unimplemented abstract methods

Potential Programmer Problems

Allows you to select the severity level (error, warning or ignore) for the following:

- Undefined local variable
- Undefined function
- Undefined field (slow check)
- Undefined method (slow check)
- Undefined class or interface
- Assignment in condition (e.g. 'if (a = b)')
- Assignment has no effect (e.g. 'a=a')
- Bad escape sequence in string

To apply Semantic Analysis Preferences settings to a specific project only:

1. Select the link labelled "Configure Project Specific Settings".
2. Select the required project from the list.
   A Semantic Analysis Properties page will appear.
3. Select the required settings and click Apply.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project.
   -Or- click No for a rebuild to be performed only when Zend Studio is restarted.
   -Or- click Cancel to cancel the operation.

Note:

Semantic Analysis settings can also be configured for an existing project by right-clicking the project in PHP Explorer view and selecting Properties | Semantic Analysis Properties.

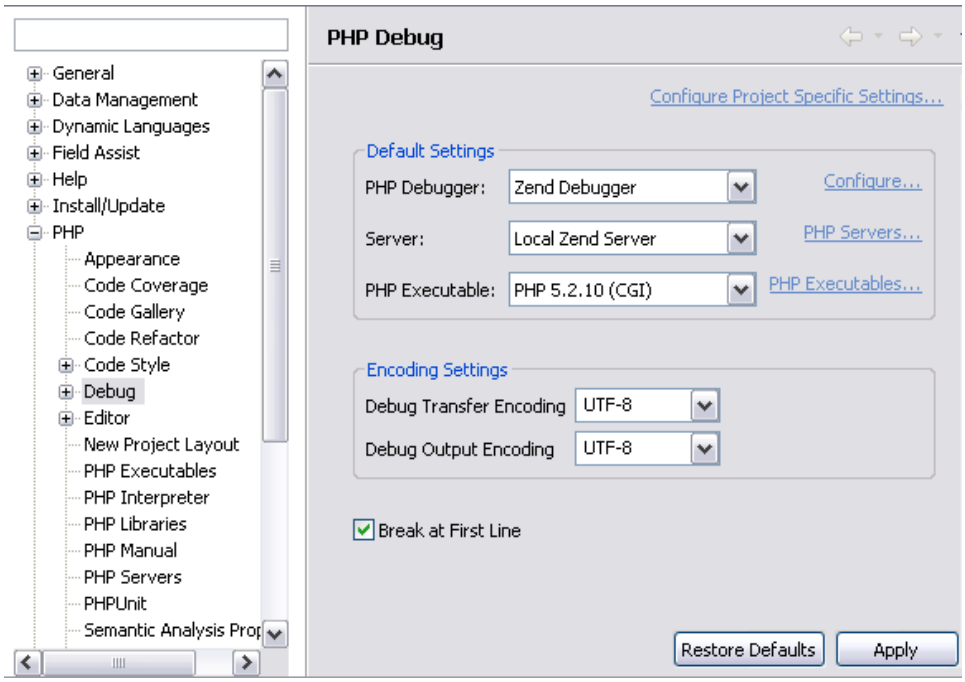
Configuring Tunneling Debug Preferences

Before initiating a debug session using tunneling, you must ensure the correct port numbers are configured in the Debug Preferences page.

This procedure describes how to configure your Debug preferences to enable tunneling.

To configure Debug Settings for Zend Studio:

1. Open the PHP Debug Preferences page by going to **Window | Preferences | PHP | Debug**.



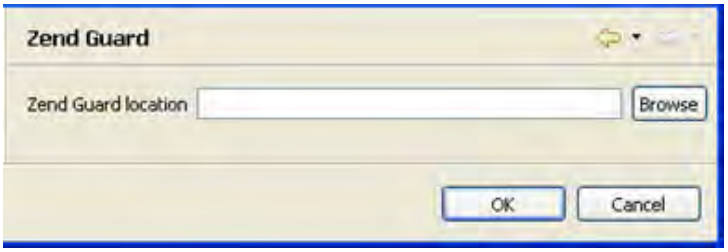
2. In the "Debug Options" section, ensure the following default settings are configured:
   - The Zend Debugger port number must be 10137.
   - In the "Advanced Zend Debugger Options" category, the Broadcast Port must be set to 20080.
   - In the Client Host/IP category, enter the Client Host/IP to which debugging results will be returned. Zend Studio for Eclipse will automatically search for and recognize the Client Host/IP, but entering a specific Host/IP will speed up the debugging process and decrease the likelihood of session time-outs.

The next step in creating a tunneling connection is to Set your Environment to be an Allowed Host.

Zend Guard Preferences

Zend Guard protects your commercial PHP 4 and PHP 5 applications from reverse engineering, unauthorized customization, unlicensed use and redistribution.
The Zend Guard Preferences page allows you to set Zend Guard's location so that it can be accessed by Zend Studio.

The Zend Guard Preferences page is accessed from Window | Preferences | PHP | Zend Guard.



To configure your Zend Guard's location:

1. Click Browse.
2. Find the location of your Zend Guard installation and click Open.
3. Click OK to apply your settings.

The connection will be made between Zend Studio and Zend Guard.
Zend Guard's functionality can now be accessed from Zend Studio by clicking the Encode Project icon on the main toolbar -or- going to Project menu and selecting Encode Project.
See the Zend Guard Integration topic for more information.

For more information on Zend Guard and to download a trial version, please visit: http://www.zend.com/en/products/guard.

PHP Project Properties

PHP Project Properties

A project's properties pages allows you to configure various settings for a specific project.

Project's properties can be accessed by right-clicking the required project and selecting Properties -or- selecting the project and from the menu bar going to Project | Properties.

The properties available will depend on the resource selected. The following properties are available for PHP projects:

- Resource Properties
- Builders Properties
- Code Style Properties
  - Code Templates Properties
  - Formatter Properties
- PHP Build Path Properties
- PHP Debug Properties
- PHP Include Path Properties
- PHP Interpreter Properties
- PHP Task Tags Properties
- Project References Properties
- Run/Debug Settings Properties
- Save Actions Properties
- Semantic Analysis
- Task Repository - For more information see the Mylyn User Guide.
  - Commit Template
- Task Tags Properties
- Validation Properties
  - HTML Syntax
  - XSLT Validation
- WikiText

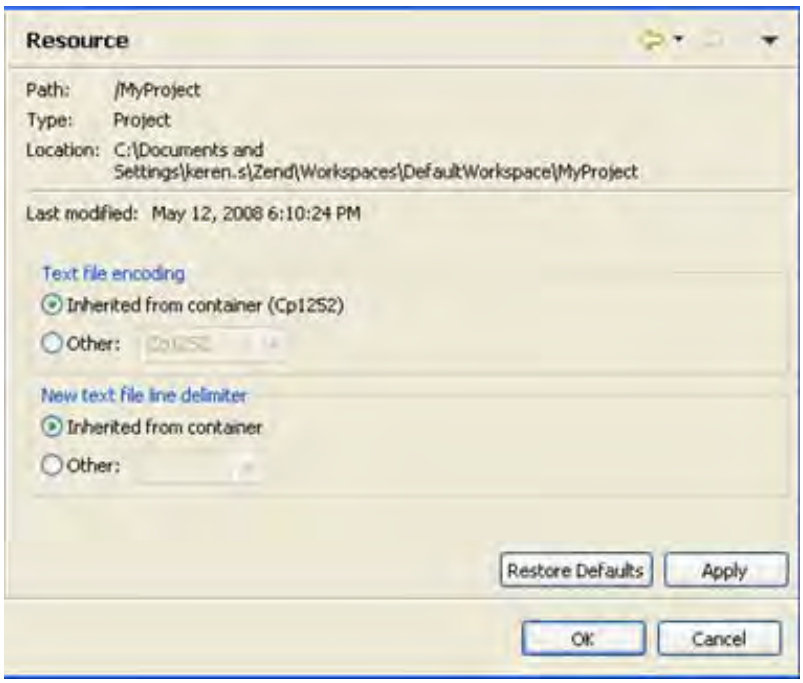
Note:

Properties pages are configured for a specific project only. To configure settings for all PHP projects, go to the PHP preferences pages (Window | Preferences | PHP).

Resource Properties

The Resource Properties page displays information about your project and allows you to set the project's text file encoding and line delimeters.

To access the Resource Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Resource.



The following information is displayed in the Resource Properties page:

- **Path**: For PHP Projects, this will be the project's name. If a file inside a project was selected, this would display the file's location within the project.
- **Type**: For PHP Projects, this will be 'Project'.
- **Location**: The project's location on the file system.

To configure PHP project resource properties:

1. Select options for:
   - **Text File encoding:**

     By default, this will be inherited from the container (this will be determined according to your local settings).

     If your text files use a different encoding scheme (because, for example, they originated from an external source), you can change the encoding by selecting the 'Other' option and choosing the required encoding from the drop-down list.

- **New text file line delimiter:**
  This will define the line delimiter for new text files. Select to inherit from container or select 'Other' and choose the required option from the drop-down list.

Note:

These settings can be configured for all newly created projects through the Workspace preferences page (Window | Preferences | General | Workspace).

2. Click Apply to apply your settings.

Builders Properties

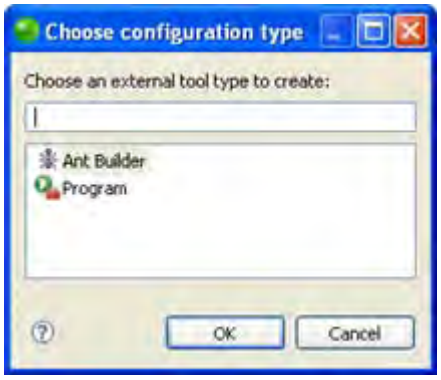
The Builders Properties page allows you to configure the External Tool builders that are run when this project is built and add, remove, or reorder external tools in the build order. The Eclipse build process scans external resources which are referenced in a project so that their contents can be made available for operations such as Content Assist and Refactoring.

To access the Builders Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Builders Properties.

By default, Builders will be added according to the type of resources in your projects (e.g. if you add JavaScript libraries the JavaScript builder will be added.).
However, you can also configure your own external builders, if required.

To add a new Builder:

1. Click New.
   The Choose Configuration Type wizard is displayed.



2. Select the required Builder type and click OK.
3. A wizard will be displayed allowing you to configure your Builder.
   See the External Tools topic in the Workbench User Guide for information on configuring your Builder.

Note:
Additional user guides can be accessed from inside Zend Studio by going to Help | Help Contents, or from the Eclipse Online Documentation site (http://help.eclipse.org/ganymede/index.jsp).

4. Click OK to apply your settings.

Code Style Properties

Code Templates Properties

The Code Templates Properties page allows you to configure the code and comments that are automatically created for different types of elements for the selected project.
See Code Templates Preferences for more information.

To access the Code Templates Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to **Project | Properties | Code Style | Code Templates**.
See

To configure Code Templates Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See Task Tags Preferences for more information on the settings available.
3. Click OK to apply your settings.

Default Code Templates Properties for all projects can be set in the Code Templates Preferences page (accessed by going to **Window | Preferences | PHP | Code Style | Code Templates** -or- by clicking the Configure Workspace Settings link on the properties page.)

Formatter Properties

Zend Studio can auto-format scripts to organize them into an easily readable format. The Formatter Properties page allows you to customize the way they are formatted for the project.

To access the Formatter Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Code Style | Formatter**.

To configure Formatter Properties for the project:

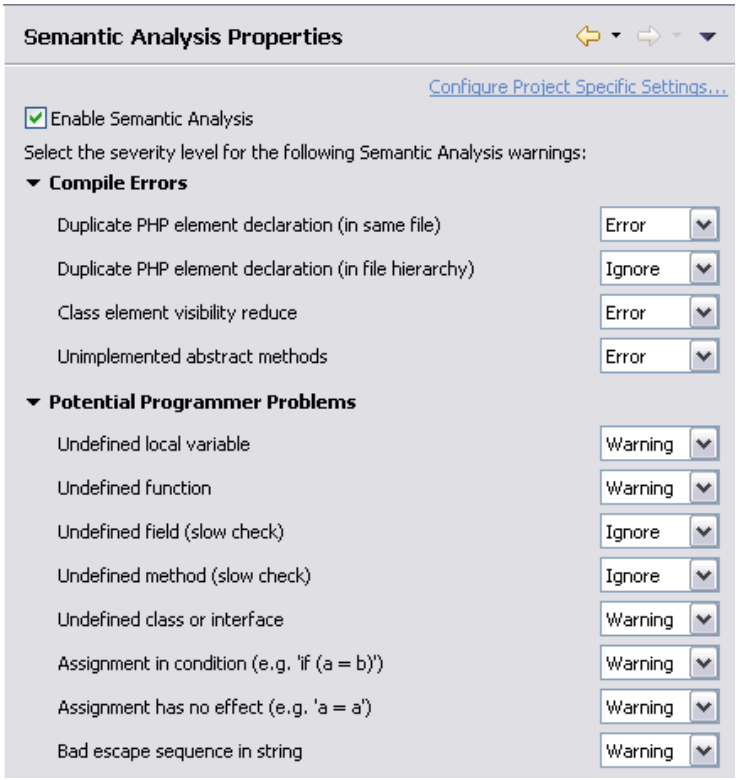
1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.
   See Formatter Preferences for more information on the settings available.
3. Click Apply.

Default Formatter Properties for all projects can be set in the Formatter Preferences page, (accessed by going to **Window | Preferences | PHP | File Network** -or- by clicking the Configure Workspace Settings link on the properties page.

Semantic Analysis Properties

The Semantic Analysis Properties page allows you to configure Semantic Analysis Properties for your project. The Semantic Analysis feature enables warning and error messages to be displayed when Zend Studio detects possible errors or problems in your script.

To access the Semantic Analysis Properties page, right-click a PHP project in PHP Explorer view and select Properties | Semantic Analysis -or- select the project and from the menu bar go to Project | Properties | Semantic Analysis.



To configure Semantic Analysis Properties for the project:

1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.
   See Semantic Analysis Preferences for more information on the settings available.
3. Click Apply.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project.
   -Or- click No for a rebuild to be performed only when Zend Studio is restarted.
   -Or- click Cancel to cancel the operation.

Default Semantic Analysis Properties for all projects can be set in the Semantic Analysis Preferences page (accessed by going to (Window | Preferences | PHP | Semantic Analysis Properties -or- by clicking the Configure Workspace Settings link on the properties page).

PHP Build Path Properties

The PHP Build Path Properties page allows you to configure the project's PHP Build Path.

To access the PHP Build Path Properties page, right-click a PHP project in PHP Explorer view and select Properties | PHP Build Path Properties -or- select the project and from the menu bar go to Project | Properties | PHP Build Path Properties.

See Configuring a Project's PHP Build Path for more information.

Formatter Properties

Zend Studio can auto-format scripts to organize them into an easily readable format. The Formatter Properties page allows you to customize the way they are formatted for the project.

To access the Formatter Properties page, right-click a PHP project in PHP Explorer view and select **Properties | Resource** -or- select the project and from the menu bar go to **Project | Properties | Code Style | Formatter**.

To configure Formatter Properties for the project:

1. Mark the Enable project specific settings checkbox.
2. Configure the settings according to your preferences.
   See Formatter Preferences for more information on the settings available.
3. Click Apply.

Default Formatter Properties for all projects can be set in the Formatter Preferences page, (accessed by going to **Window | Preferences | PHP | File Network** -or- by clicking the Configure Workspace Settings link on the properties page.

PHP Debug Properties

The PHP Debug Properties page allows you to configure default settings used when debugging files in the project.

To access the PHP Debug Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | PHP Debug Properties.

To configure PHP Debug Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See Debug Preferences for more information on the settings available.
3. Click OK to apply your settings.

Default PHP Debug Properties for all projects can be set in the Debug Preferences page (accessed by going to Window | Preferences | PHP | Debug -or- by clicking the Configure Workspace Settings link on the properties page.)

PHP Include Path Properties

The PHP Include Path Properties page allows you to configure the project's Include Path.

To access the PHP Include Path Properties page, right-click a PHP project in PHP Explorer view and select Properties | PHP Include Path -or- select the project and from the menu bar go to Project | Properties | PHP Include Path Properties.

See Configuring a Project's PHP Include Path for more information.

PHP Interpreter Properties

The PHP Interpreter Properties page allows you to set the PHP version used for the project. This will affect the internal debugger, code analyzer and content assist options.

To access the PHP Interpreter Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | PHP Interpreter Properties.

To configure PHP Interpreter Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See PHP Interpreter Preferences for more information on the settings available.
3. Click Apply.
   A prompt dialog will appear stating that a rebuild of the project must occur for the settings to take effect.
4. Click Yes to rebuild the project.
   -Or- click No for a rebuild to be performed only when Zend Studio is restarted.
   -Or- click Cancel to cancel the operation.

Default PHP Interpreter Properties for all projects can be set in the Debug Preferences page (accessed by going to Window | Preferences | PHP | Debug -or- by clicking the Configure Workspace Settings link on the properties page.)

PHP Task Tags Properties

The PHP Task Tags Properties page allows you to add new task tags and edit existing ones. Tasks are used as reminders of actions, work to do or any other action required by the programmer. Task tags are strings used by Zend Studio to recognize when tasks are added in your script. Anything after these strings inside a comment will be recognized as a task.

To access the PHP Task Tags Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | PHP Task Tags Properties.

To configure PHP Task Tags Properties for the project:

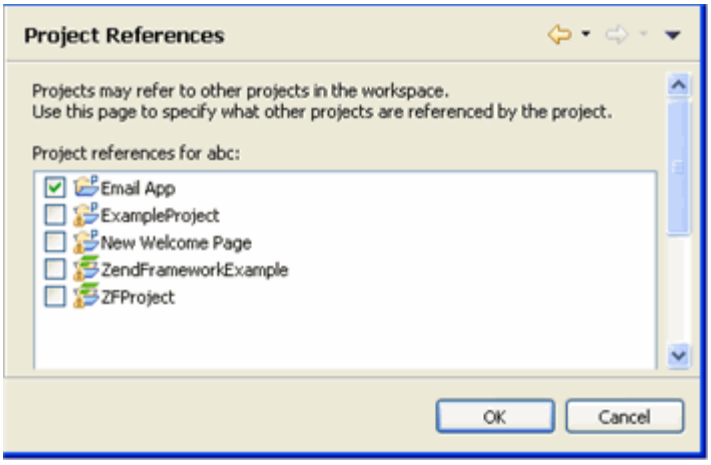
1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See Task Tags Preferences for more information on the settings available.
3. Click OK to apply your settings.

Default PHP Task Tags Properties for all projects can be set in the Task Tags Preferences page (accessed by going to Window | Preferences | PHP | Editor | Task Tags -or- by clicking the Configure Workspace Settings link on the properties page.)

Project References Properties

The Project References Properties page allows you to set the projects which are referenced by your project. This affects actions such as opening and closing of projects. (i.e. if you close a project and re-open it, you will be prompted to also open the projects which it references.)

To access the Project References Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Project References Properties.



To set the projects referenced by your project:

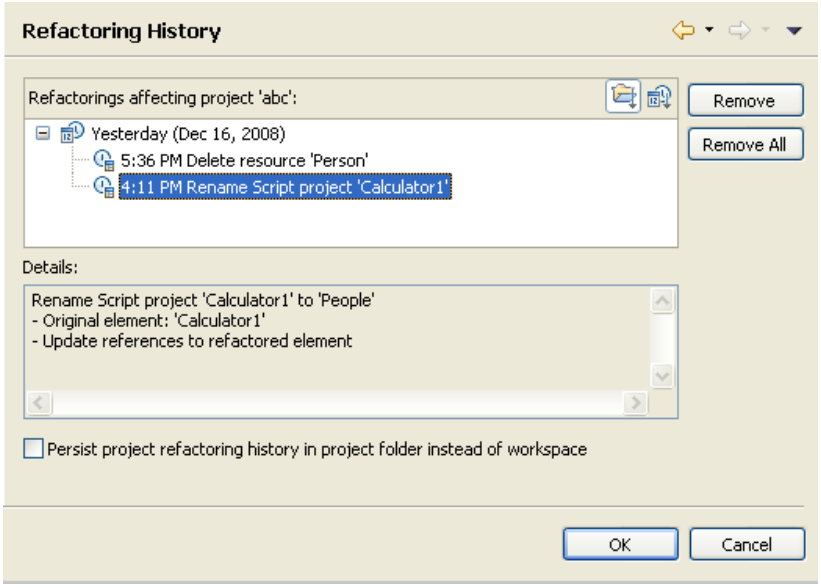
1. Mark the checkbox of the projects that are referenced within your project.
2. Click OK.

The selected projects are now specified as referenced by your project.

Refactoring History

The Refactoring History Properties page allows you to view all refactoring operations that have been performed on the project.

To access the Resource Properties page, right-click a PHP project in PHP Explorer view and select Properties | Refactoring History -or- select the project and from the menu bar go to Project | Properties | Refactoring History.



Run/Debug Settings Properties

Note:

The Run/Debug Settings Properties page is not applicable for PHP projects.

Save Actions Properties

The Save Actions Properties page allows you to remove trailing whitespace from a file each time you save it.

To access the Save Actions Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Save Actions Properties.

To configure Save Actions Properties for the project:

1. Mark the 'Enable project specific settings' checkbox.
2. Configure the settings according to your preferences.
   See Save Actions Preferences for more information on the settings available.
3. Click OK to apply your settings.

Default Save Actions Properties for all projects can be set in the Save Actions Preferences page (accessed by going to Window | Preferences | PHP | Editor | Save Actions -or- by clicking the Configure Workspace Settings link on the properties page.)

Task Tags Properties

Note:
The Task Tags Properties page is not applicable for PHP projects.

The Task Tags Properties page allows you to configure Task Tags for non-PHP files.

To access the Task Tags Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Task Tags Properties.

Task Tag properties for PHP projects should be configured from the PHP Task Tags Properties page.

Validation Properties

Note:

The Validation Properties page is not applicable for PHP projects.

The Validation Properties page allows you to specify the validators for your project.

To access the Validation Properties page, right-click a PHP project in PHP Explorer view and select Properties | Resource -or- select the project and from the menu bar go to Project | Properties | Validation Properties.

PHP Icons

The following is a list of icons representing different PHP elements which are displayed in the PHP Explorer view and in Content Assist lists:

PHP Elements

Icon	PHP Element
	PHP Project
	PHP File (included in the Build Path)
	PHP File (excluded from the Build Path)
	PHP File (containing warnings)
	PHP File (containing errors)
	Library
	Source Folder (included in the Build Path)
	Folder (excluded from the Build Path)
	PHP Class (public)
	PHP Class (default) (Only displayed in Content Assist list)
	PHP Interface
	PHP Method / PHP keyword (only displayed in Content Assist list)
	PHP Function / Method (public)
	PHP Function (private)
	PHP Function (protected)
	PHP Variable (public)
	PHP Variable (private)
	PHP Variable (protected)
	Static Constructor
	PHP Constructor / Constant
	Abstract Member
	Final Member
	Template (only displayed in Content Assist list)
	Unknown Type (only displayed in Content Assist list)
	Namespace

Zend Framework Elements

Icon	Zend Framework Element
	Zend Framework project
	Zend Controller (displayed in MVC Outline view)
	Zend Model (displayed in MVC Outline view)
	Zend View (displayed in MVC Outline view)

Other Icons

Icon	PHP Element
	PHP project connected to a repository (e.g.CVS/SVN)

Keymap

The following table displays a list of commonly used keyboard shortcuts which can be printed for quick access.

To see a full list of shortcuts and to configure the keymap, go to Window | Preferences | General | Keys.

To bring up a list of commonly used commands, press Ctrl+Shift+L in the Editor.

Show Key Assist	Ctrl + Shift + L

	Action	Shortcut	Action	Shortcut
Source Editing	Add Block Comment	Ctrl+Shift+/	Copy Lines	Ctrl + Alt + Down
	Toggle Comment	Ctrl + /	Insert Line Below Current Line	Shift+Enter
	Generate Element Comment	Ctrl + Shift + J	Delete Line	Ctrl + D
	Toggle Mark Occurences	Alt + Shift + O	Quick Fix	Ctrl + 1
	Code Assist	Ctrl + Space	Format	Ctrl + Shift + F
	Context Information	Ctrl + Shift + Space	Format Active Elements	Ctrl + I
Navigation	Open Resource	Ctrl + Shift + R	Open Views List	Ctrl + 3
	Open Type	Ctrl + Shift + T	Show View	Alt + Shift + Q, Q
	Open Type Hierarchy	F4	Previous/Next View	Ctrl + F7 / Ctrl + Shift + F7
	Open Type in Hierarchy	Ctrl + Shift + H	Previous/Next Editor	Ctrl + F6 / Ctrl + Shift + F6
	Open Call Hierarchy	Ctrl + Alt + M	Quick Hierarchy	Ctrl + T
	Open Task	Ctrl + F12	Quick Outline	Ctrl + O
	Open Method	Ctrl + Shift + M	Find Text in Workspace	Ctrl + Alt + G
	Open Selection	F3	Go to Matching Bracket	Ctrl + Shift + P
Refactor	Move	Alt + Shift + V	Extract Variable	Alt + Shift + L
	Rename	Alt + Shift + R	Extract Method	Alt + Shift + M
Execution	Debug as PHP Script	Alt + Shift + D, H	Run as PHP Unit Test	Alt + Shift + X, U
	Debug as PHP Web Page	Alt + Shift + D, W	Terminate	Ctrl + F2
	Run as PHP Script	Alt + Shift + X, H	Use Step Filters	Shift+F5
	Run as PHP Web Page	Alt + Shift + X, W	Toggle Breakpoint	Ctrl + Shift + B

Useful Links

This page includes links to commonly used online reference information. This page can be used as a bookmarks area for web links. If you have a resource that you would like to see in this area send a mail to documentation@zend.com and the link will be added for the next version.

- PHP Manual (English)- **http://www.php.net**
- Zend Framework - **http://framework.zend.com**
- Zend Dev Zone - **http://devzone.zend.com/public/view**
- Zend Forums - **http://www.zend.com/en/forums**
- PHP Certification - **http://www.zend.com/en/services/certification**
- PHP Yellow Pages - **http://www.zend.com/en/store/education/certification/yellow-pages.php**
- Support Center - **http://www.zend.com/en/support-center**
- Knowledge Base Search - **http://www.zend.com/en/support/knowledgebase.php**
- Refactoring Information - **http://www.refactoring.com**
- HTML Tutorials - **http://www.w3schools.com/html**

Support

Zend Technologies provides a wide range of resources for obtaining additional information and support, such as the Zend Support Center, the Zend Newsletter, and the Zend Developer Zone.

Zend Support Center

The Zend Support Center is a portal for information on all Zend Product related issues.
From the Zend Support Center (http://www.zend.com/en/support-center) you can access:

Zend Forums

Hosted user forums for the Zend product user community. See what other users have to say and get answers directly from the Zend Support team.
Visit: http://www.zend.com/en/forums

Zend Support Knowledge Base

The Zend Knowledge Base contains an extensive range of articles on commonly encountered problems, troubleshooting, tips and workarounds.
Search the Knowledge Base for any Zend product related issue at https://www.zend.com/en/support/knowledgebase.php.

Online Documentation

The Zend Product Online Documentation Center is an easily browseable and searchable resource for accessing the most updated information on using all Zend Products.
Visit: http://www.zend.com/en/resources/zend-documentation/

Open a Support Ticket

If you did not find the answer to your question in any of the Support resources, you can open a ticket with the Zend Support team, who will answer each ticket on an individual basis.
This can be done through https://www.zend.com/en/helpdesk/newticket.php.

Note:
More information on specific Zend Studio component errors is sometimes recorded in Zend Studio's error log. This is located in a '.log' file, created in the .metadata folder of your workspace. (e.g. C:\Documents and Settings\Bob\Zend\Workspaces\DefaultWorkspace\.metadata\.log.)
The information in this log can help the Zend Support team to diagnose a specific problem.

Zend Newsletter

Sign up for Zend PHP email updates for the hottest updates, special promotions and useful developer information.
To sign up, log in to your Zend account at https://www.zend.com/en/user/myzend, enter your email address and click Subscribe.

Zend Developer Zone

The Zend Developer Zone is the leading resource center for PHP developers, where you can learn about PHP and meet the experts.
The Zend Developer Zone features the following:

- The PHP 5 Info Center
- Articles and Tutorials
- PHP and PEAR News Weeklies
- Worldwide Job Classifieds

Visit: http://devzone.zend.com

Feedback

Send feedback, questions and comments on the Online Help and Documentation to: documentation@zend.com

Registering Your License

Once Zend Studio is installed, all its features will be available for a 30 day trial. At the end of the 30 day trial, the professional features will be disabled. These include refactoring, getter and setter generation, Zend Framework integration, profiling, PHPUnit testing, Zend Server Integration, debugging through tunneling, advanced code formatting, code galleries, and more.
To enjoy the full range of Zend Studio functionality, you should register a valid license.
To register a valid license and activate all Zend Studio features, go to Help | Register and enter your username or order #, and license key.

You can purchase a Zend Studio license from the Zend Store at http://www.zend.com/store/software/studio or from the menu bar go to Help | Purchase a License.

Index

(

(check out 252

5

5250 Bridge 166

A

Absolute 143
Activating Tunneling 356
Add CVS Repository 42
Adding an SVN Repository 50
Allowed Host 347
amf 199
amf file 199
Analyzer 123
Appearance preferences 574
Auto Detection Port 222, 353
Average Own Time 70

B

Bookmarks 113
Bookmarks view 113
bottlenecks 70
Boundry Maker 105
bracket 107
brackets 106
Breakpoint 141
breakpoints 57, 132, 515
Breakpoints view 515
Browser Toolbar 148
Build Path 144
Build Path JavaScript 467

C

Called Parameters 517
Calls Count 70
Check Out 252
Checking Out Projects 42, 50
Class Type Hints 37
Code Analyzer 123
Code Assist 37, 100, 223
Code Assist preferences 603
Code Coverage Preferences 576
Code Coverage Summary 70
Code Coverage Summary View 527
code elements 223
Code Folding 110, 227
Code Gallery 162, 423, 424, 425, 428
Code Gallery preferences 578
Code snippet 162, 423, 424, 425, 428
code tracing 154, 198
code tracing integrate 154
code tracing Zend Server 154
colors 105
comment 242
Commenting 111
Commenting Code 111
Comments 111
communication port 149
Communication Settings 149
communication tunnel 222, 353
compare 246
Concurrent Versions System 42, 125
Conditional Breakpoints 141
Configure Include Paths 379
Connecting to a Database 306
Connection Profile 304
Content Assist JavaScript 467
Controller 127, 276
Cookie 222, 353
Covered Lines 70
Current Working Directory 143
CVS 42, 125, 249, 250
CVS connection 249, 252, 256

CVS perspective 250
CVS Repository 42, 250, 252, 256
CVS Repository Exploring 250
CVS Repository Exploring Perspective ..125, 252
D
data execution 202
Data Source Explorer 306
Data Source Explorer view 301
Data Tools Platform 129, 304
database 129
Database Development Perspective 129, 301, 306, 312
debug JavaScript 467
Debug JavaScript 467
Debug preferences 593
Debug Settings 643
Debug view 512
debugger 132
Debugger Toolbar 148
Debugging 57, 132, 512
Debugging PHP Scripts 57
Debugging PHP Web Page 57
Debugging URLs 57
develop JavaScript 467
Develop with JavaScript 467
DocBlock 244
Dojo 467, 490
Dojo Integration 467
Duration Time 70
E
Easy File Creation 180
Easy PHP File 182
edit 636
edit port 636
editor 202
Editor preferences 602
Electronic Licensing solution 163
elements JavaScript 467
enable JavaScript support 467
Enable JavaScript Support in PHP Project 467
Enable Project Settings 97
encode 436
encoding 163
error workflow 154
errors 122
event file 199
execution data 154, 198, 199, 202
Execution Flow view 525
Execution Statistics 70
Execution Statistics View 523
Execution Time 70
export 199
export a Zend Server Event File 199
external file 183
external projects 143
F
Failure Trace 77
file 181, 182
file amf 199
file PHP 467
file xml 199
files JavaScript 467
Filter Stack Trace 77
Finding and Replacing 240
Firefox 148
Firewall 149
fold 110
Folding 227
Folding Preferences 605
font 105
Format Active Elements 226
Format Document 226

Formatter 226
Formatter preferences 584
Framework 127, 276
Framework Project 127, 276
Free Registration 166
FTP 266, 267, 270, 272, 502
FTP/SFTP connection 266
Function Parameter Hint 37

G

Get content 145
getters and setters 414
Goto Source 234
Guard 163, 439
Guard Preferences 645

H

HereDoc 105
Hover Support 114
Hovering 114
Hovers Preferences 606
HTML editing 422
HTML file 422

I

i5 Edition 166
i5 Edition license 166
i5 PHP API Toolkit 166
Import 192, 198, 199
import wizard 199
importing Zend Server Event File 199
Include Path 143, 379
Include Path Libraries 379
Include Paths 145
Include Statements 100
included 408
Includes 157
inserting elements 37
Installed Debuggers preferences 595
integrate 154
integrate code tracing 154
Integration Dojo 467
Internal Debugger 132
Internet Explorer 148
issue memory 154
issue memory performance 154
issue performance 154

J

JavaScript 159, 467, 468, 471, 480, 481, 485
JavaScript Build Path 467, 471
JavaScript Content Assist 467, 481
JavaScript debug 467
JavaScript Develop 467
JavaScript elements 467
JavaScript file 467
JavaScript library 467, 471
JavaScript library jQuery 467
JavaScript library Prototype 467
JavaScript Mark Occurrences 467, 487
JavaScript Open Type 489
JavaScript support enable 467
JavaScript support PHP project 467
JavaScript Syntax Coloring 467, 485
JavaScript types 467
JDBC connection profile 301
jQuery JavaScript library 467
JSDoc 467

K

keymaps 194

L

libraries 143
library JavaScript 467
listener trace 636
loading-time 70
Local Copy 132
Local Debugging 132
Local History 124, 245, 246

Locally Debugging PHP Scripts.....57

M

manage.....199, 636
manage port.....636
Manual.....120
Mark Occurrences.....108, 239
Mark Occurrences JavaScript.....467
Matching Brackets.....107
matching pair.....107
memory issue.....154
memory performance issue.....154
minus sign.....227
Model.....127, 276
Move.....67, 157, 401, 408
Moving Files.....408
MVC.....127
MVC Outline view.....127

N

NAT.....149
New PHP file.....35, 181, 182
New PHP Project.....35
news feed.....164
Number of Files.....70

O

open.....198, 202
Open Debug dialog.....57
Open JavaScript Types.....467
Open Method.....233
Open PHP Element.....232
open source.....202
open the source of trace data.....202
Open Type.....233
Organize Includes.....67, 157, 401
Others Time.....70
Outline view.....467
override / implement method.....417
Override Indicators.....116
Own Time.....70

P

parameter.....517
Parameter Stack view.....517
Parentheses.....106
Path.....70
Path Map.....350, 386
Path Mapping.....145
pattern.....106
performance issue.....154
perspective Zend Server Code Tracer.....199
PHP API Toolkit.....166
PHP Debug Perspective.....510
PHP element.....229
PHP Elements.....668
PHP Executable.....57, 97
PHP Executables Preferences.....625
PHP Explorer view.....497
PHP File.....35, 181, 182, 467
PHP Functions view.....532
PHP Icons.....668
PHP Include Path.....143
PHP Interpreter.....97
PHP Interpreter Preferences.....627
PHP Manual.....120
PHP Manual preferences.....631
PHP Preferences.....573
PHP Profile Perspective.....520
PHP Project.....35
PHP Project Outline view.....499, 534
PHP projects JavaScript support.....467
PHP Script.....70, 130, 132, 135, 315, 317
PHP Script Local Debugging.....132
PHP Script Local Profiling.....135
PHP Script Remote Debugging.....132
PHP Script Remote Profiling.....135
PHP Scripts.....57

PHP Servers Preferences 633
PHP version.....97
PHP Web Page..... 57, 130, 132, 319
PHP Web Page Debugging.....132
PHP Web Page Profiling 135
PHP/HTML WYSIWYG.....422
PHPDoc.....419
phpDoc Block Comments.....112
PHPDoc Comment 244
phpDoc comments.....112
phpDoc tags.....112
PHPDocBlock 242
PHPDocBlock comment 242
PHPDocs 160
PHPDocument1.....182
PhpDocumentor.....160
PHPUnit Preferences 638
PHPUnit Reporting 155
PHPUnit Test.....155
PHPUnit Test Case 77, 155
PHPUnit Test Reports 77
PHPUnit Test Suite.....77, 155
PHPUnit view.....77
Platform 433
Platform Integration 354
Platform Server.....433
plug-ins 96
port.....149, 636
port edit.....636
port manage.....636
preferences tracer.....636
problems.....122, 123
Problems view 122
profile 135
Profile Perspective.....520
Profiler 70
Profiler Information 70
Profiler Information View..... 522
Profiling 135
Profiling Monitor view..... 521
Profiling Perspective 70
Profiling PHP files 70
Profiling PHP Scripts..... 70
projects 467
Prototype JavaScript Library..... 467

Q

Query 70, 312
Quick Type Hierarchy View 235

R

Refactoring..... 67, 157, 401, 402, 408
Relative Path..... 143
Remote Debugging..... 132
Remote Profiling 70, 135
Remote Running..... 130
remote server..... 149, 356
Remote System Explorer..... 266
Remote Systems..... 267
Remote Systems view 502
Remotely Debugging PHP Scripts..... 57
Rename 67, 157, 401, 402
Renaming Elements..... 404
Renaming Files..... 402
Replacing Files 247
repository 42, 125, 126
Repository Location 42
required..... 408
Restoring Deleted Files 248
RSE..... 266, 502
RSS..... 164, 444
RSS feed..... 164, 444
RSS view 164
run..... 130, 315, 317, 319
Running..... 130

S

Sample Contents 304
Scrapbook 312
Searching for PHP Elements 229
security device 149
server 132
Server Debugger 57
Server Location 145
Server Path Maps 145
Servers view 505
service 441
Set the JavaScript Build Path 467
Set Up and Use Dojo Integration 467
SFTP 266, 267, 270, 272, 502
Share Project 256, 265
Sharing Projects 42, 50
Smart Goto Source 234
SOAP Client 441
source 154, 202
source control 42, 50, 125, 249
source open 202
SQL Connection 301
SQL Databases 301, 304
SQL query 312
SQL Results view 304
SQL scrapbook 312
Square 106
stack trace 512
Strings 106
Subversion 50, 126
Subversive 257
SVN 50, 126
SVN connection 257
SVN repository 258, 260, 265
SVN Repository Exploring 258
SVN Repository Exploring Perspective 126
SVN server 258
Syntax Coloring 105
Syntax Coloring JavaScript 467
Syntax Coloring preferences 612
syntax elements 105

T

Table 304
table content 306
Task Tags preferences 614
Team 256, 265
Team Environment 125
Templates 224
Templates Preferences 616
Test Case 77, 155
Test Reports 77
Test Suite 77, 155
tooltip 114
Total Execution Time 70
Total Request Time 70
Total Time 70
trace 636
trace data 154, 198, 199
trace data source open 202
tracer 636
tracer listener 636
tracer preferences 636
tracing code 636
Tunneling 149, 352, 354, 356
tunneling icon 356
Tunneling server 352
Type Hierarchy 118, 235
Type Hierarchy view 507
types JavaScript 467
Typing preferences 619

U

uncomment 242
Unit testing 77, 155
Update Manager 96

URL Debugging 132
URL Profiling.....135
Use and Configure JavaScript Content Assist.....467
Use and Configure JavaScript Mark Occurrences.....467
Use and Configure JavaScript Syntax Coloring.....467
V
Variable.....379
Variables view.....513
View 127, 276
View JavaScript Elements in the Outline View 467
view Outline 467
W
warning 122, 123
Web Page Debugging 132
Web Page Profiling.....135
Web Services Description Language 165
weblogs.....164
wizard import 199
Work with jQuery JavaScript Library 467
Work with JSDoc 467
Work with Prototype JavaScript Library...467
Workbench.....5
Workbench Options preferences.....600
workflow error 154
Working Set 237
Working Sets 117
working with.....198
WSDL 165, 441
WYSIWYG.....422
X
XML 165, 199
xml file.....199
Z
Zend 5250 Bridge 166
Zend Action Helper 294
Zend Browser Toolbar 148
Zend Code Gallery..... 428
Zend Controller 288
Zend Core 347
Zend Debugger 132, 347
Zend Debugger Toolbar..... 148
Zend Framework..... 127, 276, 279
Zend Framework Elements..... 668
Zend Framework Example Project 283
Zend Framework Project 127
Zend Guard..... 163, 436, 439
Zend Guard Preferences 645
Zend Imports..... 192
Zend Module 286
Zend Network..... 162
Zend Platform 149, 222, 347, 353, 433
Zend Platform GUI..... 222, 353
Zend Server 151, 154, 198, 199
Zend Server Code Tracer 199
Zend Server Code Tracer perspective.... 199
Zend Server Event File 199
Zend Server Event File export..... 199
Zend Studio..... 194
Zend Studio 5..... 192
Zend Studio Browser Plugin 148
Zend Studio Client Settings 222, 353
Zend Studio keymap..... 194
Zend Table..... 290
Zend Tool..... 298
Zend Toolbar..... 148
Zend View..... 291
Zend View Helper 292
ZendIEToolbar.dll..... 148
ZendPlatform 433